VERSIONE PLUS RIVISTA+LIBRO+2CD € 15,00
VERSIONE STANDARD RIVISTA+2CD € 7,70
Periodicità mensile • MARZO 2003 • ANNO VII, N.3 (67)
Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL

ioProgrammo

IN ALLEGATO

C++ Guida alla programmazione

Voice XML

Sviluppare un sito Web consultabile a voce anche tramite telefono

Al via il corso di MATLAB

SISTEMA

- Un'applicazione Java per creare e gestire file PDF
- Monitorare il BIOS con un programma Visual Basic
- Scrivere codice affidabile e robusto con JUnit

WEB PROGRAMMING

- J2EE: un'alternativa a .NET per realizzare Web Services
- .NET e SOAP per interagire con Database remoti

CORSI

- C#, l'overload degli operatori
- C++, funzioni virtuali e classi astratte
- JSP, la tecnologia JavaBean

MULTIMEDIA

- Tecniche d'illuminazione avanzata in 3D Studio Max 5

ADVANCED PROGRAMMING

- Studiare funzioni matematiche con gli algoritmi genetici

Bar Code e Visual Basic

Aggiungere al proprio gestionale funzioni per creare e gestire i codici a barre

Elettronica

Progettare un'applicazione che implementa un oscilloscopio digitale

DirectX 9

Tutte le novità per sfruttare appieno la potenza 3D delle ultime schede grafiche

Flash MX

Realizzare videoconferenze Web in maniera semplice e veloce

ISSN 1128-594X
30067
9 771128 594641

All'interno 2 CD-Rom con oltre 1.3 Gb di software

è una pubblicazione EDIZIONI MASTER

Contents

Anno VII - n. 3 (67) Marzo 2003

Respirare informatica

Il 31 gennaio Bill Gates ha presenziato ad una edizione speciale del Developer Days 2003 organizzato da Microsoft Italia. Nel prossimo numero troverete un ampio resoconto di questo evento. In questa sede vogliamo giusto segnalare uno dei punti salienti della "visione" che Gates ha presentato dei prossimi dieci anni, quelli che egli stesso ha definito "Digital Decade": la scomparsa dell'informatica.

No, non temete, non stiamo parlando della fine dell'informatica, ma del suo tendere alla trasparenza e alla più assoluta pervasività. L'idea di Gates avvicina l'informatica all'aria: la respiriamo in ogni istante, ma senza vederla e… senza pensarci! Ogni passo compiuto dall'informatica nel prossimo decennio sarò rivolto in questa direzione. La maggiore intuitività dell'interfaccia offerta dai tablet PC può essere vista come l'inizio di questo cammino, il cui termine sarà un efficace sistema di riconoscimento vocale. Un computer sempre più "intelligente" e capace di interagire con l'uomo in modi sempre più "naturali". E' bene chiarire che la visione di Microsoft non prevede l'affermazione di una tecnologia a scapito delle altre, quanto un utilizzo "in concerto" di tutte le possibilità di interazione: tastiera, penna e voce ci accompagneranno per gli anni a venire. Come l'aria, l'informatica si troverà ovunque: telefonini, orologi da polso e altri microdospositivi saranno sempre più potenti e sempre meglio connessi. La condivisione delle informazioni ed il controllo a distanza degli elettrodomestici, modificheranno il nostro vivere quotidiano più di quanto non l'abbiano già fatto. E anche in questo caso Gates afferma che il futuro sta già bussando alla nostra porta con la tecnologia SPOT che, in piccoli gadget miniaturizzati, riesce a condensare la capacità di ricevere informazioni via etere ed elaborarle con una potenza di calcolo pari a cinque volte quella dei primi PC IBM. Il tutto in oggetti non più grandi di uno Swatch! Il decennio digitale sarà insomma un'epoca di grandi trasformazioni, e i programmatori potranno giocare un ruolo chiave.

Raffaele del Monaco

ioProgrammo

Anno VII - N.ro 3 (67) - Marzo 2003 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 - Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo

Dir. Editoriale Massimo Sesti
Dir. Responsabile Romina Sesti
Product Manager Antonio Meduri
Editor Gianfranco Forlino
Coordinatore redazionale Raffaele del Monaco
Redazione Antonio Pasqua, Thomas Zaffino
Collaboratori M. Autiero, R. Bandiera, L. Buono, M. Canducci, P. Canini, S. Cristiano, F. Grimaldi, M. Del Gobbo, A. Marroccelli, S. Meschini, V. Muraglia, C. Pelliccia, N. Pepé, P. Perrotta, F. Sara, L. Spuntoni, E. Tavolaro, G. Uboldi, F. Vaccaro, R. Verre.
Segreteria di Redazione Veronica Longo

REALIZZAZIONE GRAFICA CROMATIKA Srl
C.da Lecco, zona ind. - 87030 Rende (CS)
Tel. 0984 8319 - Fax 0984 8319225
Coord. grafico: Paolo Cristiano
Coord. tecnico: Giancarlo Sicilia
Impaginazione elettronica: Aurelio Monaco

CSQ eco Certificato UNI EN ISO 14001N. 9191 CRMT

PUBBLICITÀ Edizioni Master S.r.l.
Responsabile Vendite Ernesto Redaelli
Agenti Vendita Elisabetta Februo, Serenella Scarpa, Cornelio Morari
Segreteria Ufficio Vendite Daisy Zonato
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321612 - Fax 02 8321764
e-mail: advertising@edmaster.it

EDITORE Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 8321482 - Fax 02 8321699
Sede di Cosenza: C.da Lecco, zona ind. - 87030 Rende (CS)
Amministratore Unico: Massimo Sesti
Responsabile Amministraz. e Finanza: Benedetto Celsa
Produzione e Logistica: Michele Carere
Diffusione: Alessandra Cervello
Marketing: Giuseppina Bruno, Leonardo Petrone, Antonio Meduri

ABBONAMENTO E ARRETRATI
Italia: Costo abbonamento annuale basic (11 numeri) € 84,70, Promozione Sconto 50% € 42,50. Costo abbonamento Plus (11 numeri + 6 libri) € 128,50, promozione sconto 50% € 64,50.
Estero: Costo abbonamento annuale (11 numeri) € 169,40, costo abbonamento Plus (11 numeri + 6 libri) € 257,00.
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,16 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 028321482. La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 028321699, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/ EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a: Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 20123 Milano
Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
tel.02 8321482
e-mail: servizioabbonati@edmaster.it

Stampa: Elcograf Industria Grafica (LC)
Stampa CD-Rom: Disctronics Italia (MI)
Distribuzione per l'Italia: Parrini & C S.p.A. - Roma

Finito di stampare nel mese di Febbraio 2003

Edizioni Master edita:
Idea Web, Go!OnLine Internet Magazine, Win Magazine, Quale Computer, DVD Magazine, Office Magazine, ioProgrammo, Linux Magazine, Softline Software World, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, Il CD-Rom di Idea Web, I Corsi di Win Magazine, Le Collection.

News

Borland è prima nell'acquisizione della licenza Microsoft .NET Framework SDK

Da questo accordo, un nuovo impulso per lo sviluppo rapido di applicazioni su .NET

La Borland Software Corporation ha annunciato di essere la prima azienda di sviluppo software ad avere acquisito la licenza Microsoft .NET Framework Software Development Kit (SDK) per .NET. "Borland è la prima azienda a ricevere la licenza .NET Framework: ciò costituisce un importante passo avanti per i clienti", ha dichiarato Eric Rudder, vicepresidente Developer and Platform Evangelism Division di Microsoft Corp. "Questo accordo rientra in un'ampia strategia di iniziative destinate a portare i vantaggi delle tecnologie .NET alle aziende di tutte le dimensioni. .NET Framework offre l'opportunità di creare soluzioni innovative e competitive". Grazie a questo accordo di licenza, Borland presenterà entro la fine dell'anno un nuova soluzione di sviluppo per .NET che rappresenterà una grande opportunità per tutti gli sviluppatori Borland (stimati in oltre tre milioni). "Come prima azienda ad avere acquisito la licenza per .NET Framework SDK, contiamo di estendere la nostra offerta di soluzioni aziendali per accelerare lo sviluppo di software per le più importanti piattaforme", ha dichiarato Frank Slootman, vicepresidente e direttore generale prodotti software di Borland. "Siamo impegnati a fornire la migliore soluzione di sviluppo ".NET Connected" indipendente per consentire ai clienti di velocizzare il ciclo di sviluppo applicativo". "A mano a mano che prosegue l'affermazione di .NET sul mercato, sempre più aziende chiederanno soluzioni .NET esclusive in grado di estendere competenze, investimenti e risorse esistenti", ha dichiarato Mark Driver di Gartner. "Una soluzione indipendente per la gestione del ciclo di sviluppo applicativo fornirà un'alternativa interessante per gli sviluppatori che, pur desiderando sfruttare i vantaggi di .NET Framework, vogliono mantenere un approccio ottimale con vari produttori". Questa notizia va ad aggiungersi alla recente acquisizione della software house Togethersoft da parte di Borland (per oltre 80 milioni di dollari) e ad alcune voci (non confermate) sulla volontà di Microsoft di acquisire Borland.

www.borland.it

Sun vuole rendere Java più semplice

Una piattaforma più amichevole e più lineare nei progetti di Sun

L'obiettivo di Sun è ridurre la frammentazione della piattaforma attraverso un maggior impegno nella Java community ed un contatto più diretto con gli sviluppatori ed i rivenditori. Del resto la semplicità è stata l'idea su cui si è fondato il successo di Java e riportare la semplicità al centro della piattaforma è un percorso obbligato per tutte le varianti che oggi compongono la piattaforma: J2EE (Enterprise Edition), J2SE (Standard Edition) and J2ME (Micro Edition). Per attuare questo progetto, un ruolo chiave l'uso dei layer (o astrazioni), una sorta di metadata che permetterà al kernel del runtime una migliore e connessione con l'esterno: classi, interfacce, campi e metodi potranno avere dei nuovi attributi. Il problema più grosso che Sun si troverà ad affrontare sarà quello di gestire e tenere allineati i suoi cinque principali progetti:

JXTA, JINI, Net Beans, OpenOffice.org ed il grid computing: una sfida impegnativa anche per un gigante come Sun.

www.sun.it

Metti Windows al polso!

Microsoft presenta uno SPOT per il futuro

Al CES di Las Vegas, Microsoft ha presentato i primi orologi da polso basati sulla nuova tecnologia SPOT (Smart Personal Objects Technology), appositamente realizzati per ricevere e visualizzare informazioni continuamente aggiornate attraverso un nuovo sistema di comunicazioni wireless che sarà lanciato in America il prossimo autunno. Bill Gates in persona ha annunciato importanti collaborazioni con Citizen, Fossi e Suunto, per la produzione dei nuovi orologi: i primi oggetti intelligenti che hanno come obiettivo degli effettivi vantaggi per gli utenti, e la più assoluta facilità d'uso. I nuovi orologi da polso, oltre ad essere degli avanzati cronometri (con la possibilità di cambiare aspetto e di regolarsi automaticamente sull'ora locale), hanno la capacità di ricevere e visualizzare tutta una serie di contenuti Web, dalle diverse fonti che adotteranno la nuova tecnologia wireless DirectBand di Microsoft. Una tecnologia caratterizzata da: piccole dimensioni, bassi costi, bassi consumi ed integrazione di ricevitori radio. Gli orologi da polso saranno solo gli apripista di una serie di oggetti capaci di ricevere e selezionare per noi una ricca serie di informazioni. Tra i nuovi dispositivi SPOT troveremo sveglie, portachiavi e apparecchietti magnetici da tenere sulla porta del frigo. Le informazioni saranno personalizzate e saranno legate al luogo ed al tempo. La configurazione per selezionare il tipo di informazione che ci interessa è particolarmente semplice: messaggi personali (una sorta di sms), eventi legati al calendario, news personalizzate, notizie meteo, ed informazioni finanziarie e sportive. La tecnologia SPOT è basata su una infrastruttura sviluppata da un gruppo di

ricerca Microsoft: attraverso lo sviluppo di una nuova tecnologia che integra software e hardware in un nuovo chipset, i ricercatori (lo SPOT group) sono riusciti ad ottenere una soluzione che tiene assieme i problemi relativi al consumo, al costo e alle dimensioni fisiche dei dispositivi.

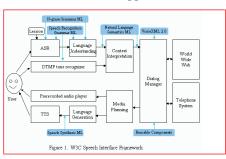
Oltre due anni di lavoro, assieme alla National Semiconductor, hanno portato alla realizzazione di un chipset comprensivo di un ricevitore a 100MHz. DirectBand di Microsoft è un insieme di tecnologie che consentono la trasmissione di informazioni Web verso i cosiddetti smat objects. Sotto l'ombrello DirectBand trovano posto un innovativo ricevitore a radiofrequenze ed un sistema wide area network basato sulla modulazione di frequenza. Entrambe queste tecnologie concorrono alla realizzazione di un efficace sistema informativo dotato di un elevato bit-rate e particolarmente robusto rispetto al rumore.

http://www.microsoft.com/resources/spot/

W3C rilascia la versione 2.0 di VoiceXML

Le nuove direttive per l'interazione vocale del Web

VoiceXML nasce con lo scopo di coniugare i vantaggi del Web e della interazione vocale nello sviluppo delle applicazioni. Le specifiche permettono agli sviluppatori di creare applicazioni vocali che racchiudono la capacità di riconoscimento vocale, sintesi vocale, comandi ad attivazione vocale, telefonia e sistemi di conferenza. VoiceXML 2.0 ha tutte le carte per modificare profondamente le applicazioni phone-base e tutte quelle che si occupano dei contatti con i clienti: al posto della antiquata interazione basata sulla pressione dei tasti sul telefono, sarà possibile ricevere e proporre informazioni per via vocale. Anche per i dispositivi multimediali di nuova generazione VoiceXML 2.0 rappresenta una grande opportunità: quella di interagire a mani libere e senza impegnare la vista in situazioni critiche, ad esempio mentre si guida. All'interno dello Speech Interface Framework, il ruolo di VoiceXML è quello di controllare le modalità di interazione fra l'applicazione e l'utente: lo *Speech Synthesis Markup Language* (SSML) si occupa di generare i prompt per gli utenti, mentre lo *Speech Recognition Grammar Specification* (SRGS) è utilizzato per guidare il riconoscitore vocale attraverso la specifica di alcune regole grammaticali. Il supporto per le funzioni di telefonia è fornito dal *Voice Browser Call Control* (CCXML) mentre il *Semantic Interpretation for Speech Recognition* definisce la sintassi e la grammatica per la sezione SRGS. VoiceXML ha molti punti in comune con lo *Speech Application Language Tags* (SALT) recentemente annunciato e che si trova attualmente al vaglio del W3C. SALT è un insieme di estensioni ai linguaggi di markup esistenti (rivolte principalmente a HTML e XHTML) atte a integrare il supporto per l'accesso telefonico e multimodale alle informazioni, ad applicazioni e Web

Figure 1. W3C Speech Interface Framework

Services. SALT si presta a essere utilizzato da un vasto schieramento di dispositivi che va dai PC ai PDA wireless, dai telefoni ai tablet PC. Sebbene distinti, VoiceXML e SALT condividono obiettivi molto simili che tendono, in alcuni casi, a sovrapporsi: SALT, ad esempio, utilizza dei componenti chiave dello the Speech Interface Framework, tra cui lo SRGS e lo SSML.

www.w3.org/TR/voicexml/

Il grid computing incontra i Web Services

La versione beta del Globus Toolkit 3.0 programmata per aprile

La tabella di marcia per l'atteso toolkit è decisamente serrata: in aprile dovrebbe essere rilasciata la prima beta, subito seguita dal rilascio della versione ufficiale, in giugno. Il Globus Toolkit 3 implementerà una architettura aperta che rappresenterà la convergenza fra il Grid computing ed i Web Services, a completamento di uno sforzo che ebbe inizio con l'alleanza fra IBM ed il progetto Open Source "Global Project". Tra le caratteristiche si segnalano la *SOAP Message Security*, una serie di API e di tool per semplificare lo sviluppo di client e server OGSI, una serie di servizi per i database, il servizio GRAM (che include l'insieme di protocolli per il resource-information-discovery), il servizio GridFTP, il servizio Reliable File Transfer (RTF), il Replica Location Service (RLS) ed una serie di API per lo sviluppo dei client.

www.globus.org

IBM aiuterà AMD nella costruzione di nuovi chip

Una alleanza per ridurre i rischi connessi allo sviluppo dei processori

La collaborazione dei due giganti prenderà corpo con la produzione di processori con tecnologia a 65 nanometri (previsti per 2005) e a 45 nanometri. (pronti nel 2007). Questo accordo segue una serie di altri che hanno modificato la geografia dei produttori di chip: da un insieme di produttori indipendenti ad una serie di alleanze. La principale sfida che devono affrontare i produttori di chip è quella di ridurre la quantità di calore prodotta dai milioni di transistor presenti su ogni singolo chip e distribuire i costi della ricerca tra più aziende è diventato pressoché obbligatorio. L'alleanza tra IBM e AMD si focalizzerà principalmente sul miglioramento delle tecnologie di risparmio energetico e nella ricerca di nuovi materiali isolanti per i processori. In particolare, le tecniche di risparmio energetico hanno un immediato riflesso nelle prestazioni dei chip, essendo proprio il calore irradiato uno dei principali limiti all'aumento della velocità di clock. Questo nuovo accordo rappresenta la fine della precedente alleanza fra AMD e la United Microelectronics, una industra di Taiwan specializzata nella produzione di chip per conto terzi. La produzione dei nuovi chip, pur non escludendo la partecipazione della United Microelectronics, richiederà nuovi accordi e, tra i possibili produttori, AMD ha indicato proprio IBM.

www.amd.com

Il Software sul 1° CD-ROM

Allegro CL 6.2

Uno dei più diffusi ambienti per la programmazione Object Oriented in ambiente enterprise. Ideale per sviluppare applicazioni robuste e mission-critical, utilizzando Lisp in un ambiente rivolto alla progettazione di applicazioni Web e già pronto per i nuovi standard di comunicazione, XML in testa. Allegro si presenta come la soluzione ideale per applicazioni di data mining, gestione della conoscenza e integrazione di strutture dati preesistenti. Particolarmente interessante l'implementazione della tecnologia Dynamic Objects, che consente di creare applicazioni su cui è possibile effettuare l'upgrade mentre l'applicazione stessa è avviata.

acl62_trial.exe

Aqua Data Studio v1.5

Un potente tool per amministratori di database che consente di editare e eseguire script SQL, oltre che di ***** un'agevole navigazione nelle strutture dei più complessi database. Aqua Data Studio mette a disposizione degli utenti un potente ambiente di sviluppo integrato che può fare da interfaccia a tutti i principali database presenti sul mercato, consentendo l'esecuzione di più operazioni simultaneamente, su più database e attraverso un ambiente coerente e ben strutturato. Degno di menzione risulta essere il Query Analyzer che mette a disposizione un editor con un syntax highlighting studiato specificamente per gli RDBMS, e con avanzate funzioni di auto-completamento che velocizzano notevolmente il lavoro degli sviluppatori. La possibilità di analizzare per via grafica la struttura dei Database, consente una più semplice interpretazione dei dati e delle correlazioni. Aqua Data Studio può salvare i risultati delle query in numerosi formati, compresi HTML e XML. Gratuito.

adstudio.exe

COM-X-Ray 1.2

Una utilissima applicazione per chi sviluppa componenti COM: sarà più semplice effettuare il debug e ottimizzare le nostre applicazioni attraverso una approfondita conoscenza delle interazioni fra i vari componenti dell'applicazione stessa. È possibile studiare la struttura interna di un oggetto COM e stampare un diagramma delle dipendenze esterne che lo legano ad altri oggetti. Versione di prova valida 30 giorni.

COM-X-RAY.exe

Ghost Installer Studio v3.4

Con Ghost Installer è possibile creare dei file dei pacchetti di installazione contenuti completamente in un singolo file. Il kit di sviluppo consente di personalizzare approfonditamente l'interfaccia del setup e offre il supporto multilingua oltre a fornire automaticamente un programma di disinstallazione. Per realizzare pacchetti assolutamente professionali non è richiesto alcuna conoscenza di programmazione: tutte le personalizzazioni e le istruzioni possono essere impartite per via visuale. Grazie all'integrazione di un potente algoritmo di cifratura e alla possibilità di associare un serial number per ogni utente, Ghost Installer garantisce una più efficace protezione dalla copie illegali del software che distribuiamo. L'interfaccia utente di Ghost Installer Studio permette di inviare una chiave di registrazione ad ogni utente che ne faccia richiesta: il tutto è gestito per via visuale e senza alcun intervento da parte dello sviluppatore. Anche la generazione e l'archiviazione delle chiavi relative ad ogni utente sono operazioni gestite in modo del tutto automatico da Ghost Installer. Più che una singola applicazione, Ghost Installer può essere visto come una vera e propria piattaforma, grazie al supporto di plug-in e di un API appositamente dedicata. In questa nuova versione è stata raggiunta la piena compatibilità con la piattaforma .NET di Microsoft, oltre a numerose ed interessanti funzioni come la possibilità correggere installazioni corrotte e la funzione di Rollback.

GIStudioDemo.exe

MDB2PDF PDF Converter 1.0

Un tool utilissimo e che non dovrebbe mancare nella cassetta degli attrezzi di un amministratore di database.
Con MDB2 PDF è possbile convertire in formato PDF qualsiasi oggetto appartenente ad un database Access: Report, Form, Tabelle e Queries. Con pochi clic è possibile generare il corrispettivo in PDF del risultato che avremmo a schermo.
Versione dimostrativa, su ogni pagine è aggiunta una scritta che ne rende inutilizzabile l'output.

mdb2pdf-demo.exe

JAVA

Web Services Developer Pack 1.1_01

Un insieme di tool che, insieme alla piattaforma Java, consentono agli sviluppatori di costruire, testare e pubblicare un vasto insieme di software diversi: applicazioni XML, Web Services e applicazioni Web. Il Java Web Services Developer Pack (o WS DP) include le implementazioni standard dei più diffusi protocolli per Web Services: WSDL, SOAP, ebXML e UDDI. Sono inoltre presenti delle implementazioni fondamentali per lo sviluppo di applicazioni Web come Java Server Pages, e la libreria standard di tag JSP. Questi standard Java consentono agli sviluppatori di inviare e ricevere messaggi SOAP, cercare e recuperare informazioni in registri UDDI ed ebXML, oltre a d permettere la rapida costruzione e la pubblicazione di applicazioni Web basate sui più aggiornati standard JSP. Il Java Web Services Developer Pack v1.0_01 include:

- Java API for XML Messaging (JAXM) v1.1_01,
- Java API for XML Processing (JAXP) v1.2_01,
- Java API for XML Registries (JAXR) v1.0_02,
- Java API for XML-based RPC (JAX-RPC) v1.0_01,
- SOAP with Attachments API for Java (SAAJ) v1.1_02,
- JavaServer Pages Standard Tag Library (JSTL) v1.0.1,
- Java WSDP Registry Server v1.0_02, Web Application Deployment Tool,
- Ant Build Tool 1.4.1, Apache Tomcat 4.1.2 container.

jwsdp-1_0_01-windows-i586.exe

Il Software sul 2° CD-ROM

ColdFusion MX

ColdFusion MX offre una serie di potenti funzioni in un ambiente di scripting server-side facili da apprendere e notevolmente produttive. L'architettura sicura, oltre ad un set completo di funzioni incorporate, si traducono in alte prestazioni e scalabilità. Supporta tutti gli standard aperti e si integra con facilità con le strutture tecnologiche esistenti. Il linguaggio intuitivo, basato sui tag, richiede poche righe di codice per gestire automaticamente le funzioni di programmazione di basso livello e semplificare il riutilizzo del codice. La tecnologia ActionScript consente agli sviluppatori Macromedia Flash di usare lo stesso linguaggio di scripting per la logica client e server.
Le nuove funzioni di ColdFusion MX sono pienamente contemplate all'interno dell'ambiente di sviluppo di Dreamweaver MX: layout e creazione di prototipi visivi, un miglioramento delle funzioni di modifica e sviluppo di codice, funzioni di debugging integrato. L'estensibilità è garantita grazie a librerie di tag personalizzate, componenti riutilizzabili, Java/C++ e migliaia di add-on di altre marche. ColdFusion si integra perfettamente con i principali standard Internet e i modelli di componenti, tra cui XML, i servizi Web, Java, .NET/COM e CORBA. Sono supportati anche lo sviluppo e la distribuzione di applicazioni su server ColdFusion MX autonomi o su server applicativi Java più affermati, quale IBM WebSphere Application Server.

coldfusion-60-win-en.exe

Flash Communication Server MX

Con questo prodotto Macromedia, è possibile creare audio e video che possano essere utilizzati, in tempo reale, sul proprio sito web o nelle applicazioni dinamiche. Si possono realizzare presentazioni che integrano audio, video, testo, chat, o che contengano streaming video e contenuti multimediali sincronizzati, ecc…
Si possono anche creare e distribuire funzioni di comunicazione, con propri contenuti, grazie all'impiego del Flash Player che ne permette la fruizione direttamente via Browser o in modalità stand-alone.
Si possono anche aggiungere funzioni interattive per realizzare trasmissioni video e di dati, immagini condivise, aule conferenze virtuali, bacheche digitali, ecc…

FlashCommunicationServer.exe

Sun ONE Studio Community Edition 4 + Java 2 SE 1.4.1

In un unico pacchetto di installazione, potete trovare il JDK di Sun nella versione 1.4.1 insieme al Sun ONE Studio 4 (prima noto come Forte). Con un'unica installazione avremo dunque tutto quello che ci serve per sperimentare la visione Java della programmazione enterprise e dei Web Services. Il JDK incluso nel pacchetto non ha bisogno di presentazioni: l'ambiente di sviluppo Sun che negli ultimi anni si è imposto come la prima scelta per i programmatori che lavorano in ambito multipiattaforma. In questa nuova release troviamo grandi miglioramenti sul piano delle performance e della scalabilità.
Tra le tante novità della Java 1.4 si segnalano le notevoli migliorie per tutto ciò che riguarda l'IO: buffer, gestione delle regular expression, socket, channels e molto altro ancora. La minor release ha migliorato il supporto ai Web Services, la Virtual Machine dispone di due nuovi Garbage Collector e sono stati risolti oltre 2000 bug che affliggevano la precedente release. Sun ONE Studio 4 (update 1) rappresenta invece l'ultima release dell'IDE che Sun ha scelto e sviluppato appositamente per il suo gioiello.
Indirizzato agli sviluppatori più esperti, Sun ONE Studio 4 consente di realizzare applicazioni di livello professionale e permette di partire sempre con il piede giusto grazie ai numerosi template che spaziano dalla semplice classe, ad un'applicazione JSP, dai javabean ai documenti XML. Risulta molto curata anche la sezione dedicata alla interfaccia utente: grazie a Sun ONE risulta finalmente semplice costruire interfacce complete e funzionali, sfruttando a fondo sia le librerie Swing sia Awt.
L'ambiente a multifinestra è quanto di più completo si possa desiderare anche se, per sfruttare a fondo le capacità dell'IDE, sono necessarie risoluzioni abbastanza elevate.

j2sdk-1_4_1-s1studio_ce-4u1-bin-windows1.exe

JBoss 3.0.5

JBoss è un application server Open Source scritto interamente in Java. Grazie a JBoss è possibile far girare componenti EJB (Enterprise Java Beans) e qualsiasi applicazione sviluppata attraverso la tecnologia J2EE di Sun. Grazie ai connettori JMX, JBoss consente una notevole flessibilità e si presta agli utilizzi più disparati. In particolare è possibile sviluppare un proprio transaction-manager ed un proprio persistence-manager.

JBoss

ME RemoteNet 2.0

ME RemoteNet permette di lavorare su PC da un altro PC a distanza, sia tramite connessione in rete locale, sia attraverso una connessione internet. Il programma permette di ottenere la schermata del PC distante, permettendo al mouse e alla tastiera di prenderne il controllo. In questo modo, potrete lavorare sul PC come se foste sopraposto.

mecrmvr2X.exe

XML Spy 5.0 Ent. rel. 3

Il più avanzato strumento di sviluppo per XML. Uno dei punti di forza risiede nella possibilità di switchare velocemente fra quattro differenti viste per ogni documento: Enanced Grid per il dettaglio di tutti gli elementi: Schema Design; Text View per una programmazione a più basso livello, e una vista di preview che anticipa come il documento sarà visualizzato da un comune browser. E' possibile sviluppare sia documenti XML che DTD che saranno poi pubblicabili via FTP attraverso lo stesso XML Spy. Davvero eccezionale il supporto offerto ai Web Services grazie ad apposite interfacce per i protocolli SOAP, XSL e WSDL. Tra le nuove funzioni disponibili in questa release si annoverano un nuovo debugger per XSLT, un generatore di documenti WSDL, la generazione di codice C# (che si va aggiungere a quelle già disponibili per Java e C++) e l'integrazione con Oracle XML DB.

XMLSPYEntComplete5.exe

Zend Studio 2.5 Trial

Un valido supporto per lo sviluppo di applicazioni web in PHP.

Lo **Zend Development Environment** è nato per aiutare il webmaster nell'intero ciclo di sviluppo di applicazioni web basate sul linguaggio PHP, riducendo la stesura di codice, aumentandone l'accuratezza, effettuandone il debug e fornendo un valido e folto supporto di help per il linguaggio PHP. Questa eccezionale piattaforma di sviluppo risulterà utile sia per gli sviluppatori principianti sia per quelli più esperti, permettendo loro di scrivere codice e testarlo in maniera efficiente, riducendo i tempi di produzione.

Fig. 1: Il meccanismo di funzionamento dei componenti Zend nella fase di debug.

I COMPONENTI

L'ambiente si sviluppo si basa su tre componenti principali:

- PHP/Zend Engine
- Zend Debug Server
- Zend Development Environment Client

Il *PHP/Zend Engine* è il motore del linguaggio di scripting PHP. Questo componente è incluso nell'installazione della versione **4.0** di **PHP**. Lo **Zend Debug Server**, invece, è un'applicazione separata che viene usata per il debug delle applicazioni. In questa fase, infatti, il Debugger risponde ai controlli del Client e dirotta le richieste al motore di Zend. Gli *Zend Development Environment /PHP Components* sono installati in locale mentre il Debug Server viene installato su un server remoto, come si evince dalla tabella di cui sopra.

Componente	Locale	Remoto
Environment Client	Sì	No
Debug Server	Sì	Sì
PHP/Zend Engine	Sì	Sì

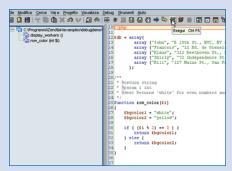
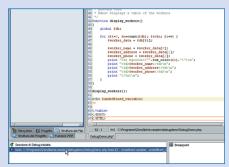
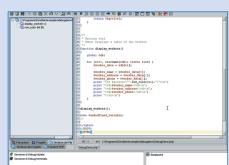
L'AMBIENTE

In **Fig. 1**, possiamo visualizzare il meccanismo di funzionamento dei vari componenti, che le lavorano insieme nella fase di debug, partendo da un comando inoltrato dal Client. In particolare, l'interazione illustrata nello schema procede secondo i seguenti passi:

1. Mentre viene eseguito il debug di uno script, la richiesta (ad esempio un breakpoint) è inoltrata dal Client e trasmessa al Debug Server.

2. Il Debug Server inoltra i comandi al motore PHP che esegue le richieste e restituisce le informazioni. Nel caso di un breakpoint, ad esempio, l'esecuzione dello script viene interrotta e vengono generate le risposte.

3. Il motore PHP restituisce le informazioni di output, variabili, call stack ed errori

Eseguiamo il debug di un'applicazione

1 Avviamo la fase di debug cliccando sull'icona opportuna. Facciamo eseguire step successivi all'applicazione per testarne tutte le sue funzioni. Se tutto va bene, la finestra segnala la fine del debugging.

2 Il Debugger ci segnala un errore. Clicchiamo sulla riga, nella finestra di debug. L'editor ci porta direttamente sul codice, esattamente nel punto in cui si è verificato il problema, per farci apportare le modifiche.

3 Modifichiamo il codice, in base ai messaggi di errore che ci vengono segnalati dal debugger. Effettuiamo di nuovo il debug fino a che la finestra non segnali la fine della fase di debug senza altri errori...

al Debug Server.

4. Il Debug Server riceve le informazioni e le trasmette (o restituisce) all'Environment Client, che si incarica di visualizzarle nella finestra opportuna…

L'ENVIRONMENT CLIENT

Lo **Zend Development Environment Client** è un ambiente integrato di sviluppo per applicazioni PHP. In particolare, l'ambiente aiuta l'utente nello sviluppo grazie all'utilizzo dei seguenti strumenti:

- **Debugging**. Controlla il testing delle applicazioni PHP fornendo informazioni per la valutazione di errori, di call stack, variabili e output.
- **Project e File Management.** Permette di amministrare documenti e cartelle all'interno dell'ambiente, incluso un gestore FTP.
- **Project Inspector e File Inspector.** Per l'esplorazione e la visualizzazione delle classi e dei loro membri.
- **Completamento del Codice.** Offre una grande varietà di completamento di codice PHP, HTML, Classi, variabili membro, variabili, parole chiave, oggetti, ecc…
- **Highlighting di syntax.** Colora automaticamente il testo secondo la sintassi degli elementi.
- **Indentazione.** Applica un'indentazione standard o personalizza per formattare il codice PHP.
- **Help on line.** Tutta la documentazione sulle funzioni PHP semplicemente premendo un tasto.
- **Navigazione Go to.** Offre diversi metodi per recarsi direttamente sulla parte di codice che interessa.
- **Personalizzazione dei tasti rapidi.** Permette di impostare numerose funzioni attraverso la personalizzazione di combinazioni di tasti e mouse.

LO ZEND DEBUG SERVER

Questo componente non fa altro che rispondere ai controlli dello Zend Development Environment Client. Con tali controlli, il Debug Server richiede al motore PHP per fermare o mettere in pausa un'applicazione PHP. Quando il Debug Server ferma l'esecuzione di un'applicazione, restituisce il report delle *variabili* e dei *call stack* all'Environment Client. Se richiesto dal Client, può anche cambiare il valore delle variabili prima che l'esecuzione venga ripresa.

Fig. 2: L'interfaccia dello Zend Development Environment Client.

FLUSSO DI SVILUPPO

Il ciclo di sviluppo di un'applicazione segue un semplice diagramma di flusso. Anche se il flusso è semplice, in generale i processi possono richiedere numerosi cicli prima che l'applicazione possa ritenersi completata. Grazie all'Environment Client, è possibile testare ed eseguire il debug di ciascuna funzione, appena questa viene scritta o dopo che l'intera applicazione sia stata completata. In **Fig. 3** è schematizzato il flusso di sviluppo per la creazione di applicazioni. Ad ogni modo, il flusso di sviluppo del codice può essere riassunto nelle seguenti fasi:

Scrittura

Gli strumenti dello Zend Development Environment, come il completamento del codice, l'indentazione, il Project Management, il Syntax Highlighting, ecc…, assistono nella scrittura del codice. Ogni volta che uno script viene completato, è possibile visualizzarne il funzionamento.

Test e Debug

La fase di testing e quella di debugging si rende necessaria per scovare qualsiasi problema dell'applicazione.

Editing

La fase in cui l'applicazione viene scritta e modificata. A questo punto, si apportano le correzioni usando le informazioni ottenute durante la fase di debug. Si possono utilizzare gli stessi strumenti utilizzati durante la fase di scrittura per velocizzare il processo. Naturalmente, la fase di editing va seguita da un'ulteriore fase di debug, al fine di eliminare qualsiasi possibilità di errore. Questo processo va ripetuto diverse volte fino ad ottenere un risultato soddisfacente.

Upload

La pubblicazione vera e propria dell'applicazione PHP/HTML. Tutti i file vengono trasferiti al server web. Questa fase viene svolta quando si ritiene che l'applicazione venga eseguita correttamente. L'upload può essere effettuato utilizzando il gestore FTP messo a disposizione dall'Environment Client.

Non resta che invitarvi a testare questo strepitoso ambiente di sviluppo, che non mancherà di stupirvi, per lo sviluppo delle vostre applicazioni PHP. Nel CD-Rom allegato, trovate le versioni trial a 21 giorni.

Eclipse 2.0.2

Nuova prospettiva sulla programmazione

Un ambiente di sviluppo Open Source potente e flessibile che, grazie ad un efficace insieme di API, consente una facile integrazione di nuovi componenti e Plug-In forniti da terze parti. Il progetto Eclipse nasce in IBM che, dopo uno sviluppo costato diversi milioni di dollari, ha deciso di "donarlo" alla comunità open source, con il dichiarato intento di andare a contrastare il Visual Studio .NET di Microsoft ed il Sun ONE Studio. Hanno contribuito al suo sviluppo: Rational, Borland, Merant, RedHat, Suse e Sybase. In questa eccellente piattaforma di sviluppo, risulta centrale il concetto di perspective (prospettiva): una prospettiva è un insieme di finestre (navigator, editor, lista delle attività, ecc.) utili nel loro insieme ad agire sul progetto corrente in diverse modalità. In pratica, a seconda della fase in cui siamo coinvolti, avremo a disposizione l'insieme di finestre più adatto alle operazioni che dovremo compiere. Sulla sinistra sono presenti dei piccoli pulsanti, a ciascuno dei quali corrisponde una particolare perspective: Resource, è la prospettiva base e permette di accedere velocemente a tutte le risorse coinvolte nel progetto; la prospettiva Java, offre invece la possibilità di navigare la gerarchia delle classi e dei pacchetti appartenenti al progetto; Debug, comprende un insieme di finestre che permettono di tracciare il comportamento di un'applicazione. Per ogni tipo di risorsa, avremo un differente ed appropriato editor che ci permetterà di interagire con essa. Come in ogni moderno IDE, per molte delle più comuni azioni avremo uno specifico wizard. Benché utilizzabile per sviluppare progetti in qualsiasi linguaggio, grazie al Java Development Tooling (JDT) incluso nell'SDK, Eclipse diviene un formidabile strumento di progettazione Java. L'editor, in particolare, offre tutte le caratteristiche più importanti come

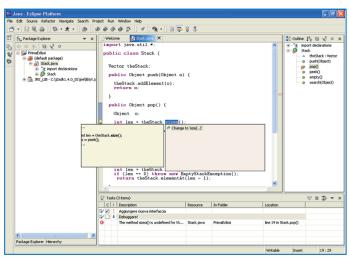
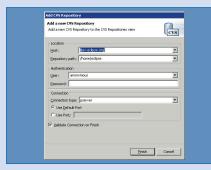
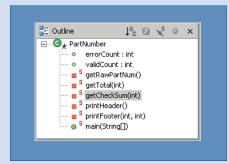
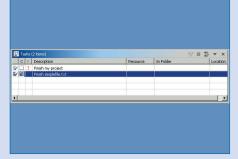



Fig. 1: **Il suggeritore contestuale in azione.**

l'autocompletamento e l'evidenziazione sintattica del codice. La caratteristica meglio implementata è quella che consente di ricavare tutte le informazioni su un oggetto, una variabile o un metodo semplicemente stazionando qualche secondo sull'oggetto stesso. Rimarcabile risulta l'integrazione con JUnit per l'esecuzione automatica di test su particolari porzioni di codice: attraverso dei semplici wizard, è possibile definire singoli test o intere collezioni. Chi ha già sperimentato Visual Age non potrà che apprezzare questa "reincarnazione" open source, per tutti gli altri ci vorrà forse un po' di tempo per abituarsi al tipo di approccio alla programmazione proposta da IBM… ma col tempo saranno ripagato con gli interessi! Nel CD-Rom: *eclipse-SDK-2.0.2-win32.zip.*

Alcuni elementi dell'ambiente

1 **Lo sviluppo in team può essere efficacemente attuato grazie al supporto per l'accesso ai repository CVS.**
Ogni programmatore opera su delle copie private del progetto, copie che possono essere sincronizzate con il repository, in ogni momento e con numerose opzioni.

2 **La Outline View è correlata alla finestra di editor attiva: a seconda del tipo di documento aperto, mostra gli elementi strutturali presenti nel file e la gerarchia degli stessi. Nell'esempio , la struttura di un sorgente Java, i cui elementi strutturali sono classi, campi e metodi.**

3 **La finestra dei task rappresenta un po' il diario di bordo dello sviluppo di un progetto.**
Le voci che vanno a comporre la vista possono essere sia automaticamente prodotte dall'ambiente (errori, warning, ecc.) sia introdotte manualmente dagli sviluppatori.

IntelliJ Idea 3.0

Uno dei più apprezzati ambienti per lo sviluppo di applicazioni Java

In questa nuova versione, aggiunge nuove importanti caratteristiche: pieno supporto per JSP/EJB, integrazione con Ant e JUnit, supporto per XML e una cospicua collezione di API per la realizzazione di plug-in. Inutile sottolineare che sono presenti tutte le più importanti caratteristiche comuni in un moderno IDE: un ottimo debugger, avanzate funzioni di search&replace, colorazione sintattica. Una menzione particolare la merita il completamento automatico del codice che si fa apprezzare con l'uso per la sua semplicità e per la flessibilità con cui segue le abitudini dello sviluppatore. Un'assenza di peso risulta essere quella di un apposito strumento per la realizzazione dell'interfaccia utente. Per poter effettuare una corretta installazione del software, è necessario collegarsi all'indirizzo *http://www.intellij .com/idea /evaluate.jsp*, al fine di ottenere una licenza di valutazione.

Il concetto di flessibilità è alla base di IDEA 3.0.E' possibile personalizzare qualsiasi aspetto dell'applicazione: dalla formattazione del codice, alla colorazione sintattica, all'organizzazione dell'import, fino ad arrivare ai colori utilizzati per evidenziare gli errori e all'orientamento delle finestre e delle toolbar. Le personalizzazioni possono essere applicate a singole porzioni di codice o a interi file, o ancora ad tutti i file di un progetto.
Inoltre, nel momento in cui importiamo del codice, ad esso viene applicata la corretta formattazione scelta per il progetto. I progettisti di IDEA, hanno poi dedicato particolare attenzione al supporto per il refactoring (il processo che prevede un continuo e costante miglioramento del codice prodotto e della sua struttura). Tutte le difficoltà che hanno spesso allontanato gli sviluppatori da questa preziosa tecnica si affrontano più semplicemente, grazie ai numerosi pattern supportati da IDEA:

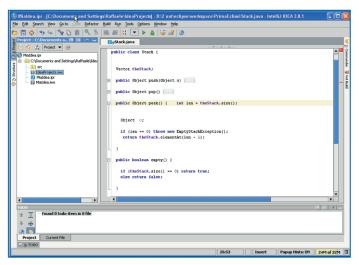
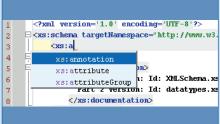
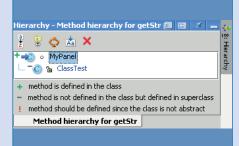



Fig. 1: Uno screenshot dell'ambiente di sviluppo.

- Cambiare la firma dei metodi per includere nuovi argomenti
- Rinominare o rimuovere classi, metodi e membri
- Introdurre una nuova variabile a partire da una specifica espressione
- Incapsulare i riferimenti ai campi in appositi metodi
- Spostare i membri nelle super classi

Con IDEA diventa più facile rinominare una classe o spostarla in un diverso package: in questi casi, IDEA non si occupa solo di effettuare gli opportuni cambiamenti al codice ma corregge tutte le istruzioni di input, gli script JSP, i documenti Javadoc e i file XML coinvolti nel progetto. Nel CD-Rom: *idea-3.0.1v2.exe*

Strumenti che assistono lo sviluppatore

1 Code inspection, il tool per l'ispezione del codice, consente di cercare facilmente le inconsistenze presenti all'interno del codice e permette di scoprire eventuali violazione delle specifiche EJB, l'utilizzo di codice deprecato, ecc.

2 Idea si dimostra un ottimo strumento per la creazione di documenti XML: le funzioni di completamento automatico del codice ed il rilevamento degli errori sono quanto di più completo si possa trovare in un editor moderno.

3 La finestra di ispezione per la struttura dei metodi consente di tenere sotto controllo la gerarchia delle classe in cui un metodo è definito (o implementato). Utilissima risulta la possibilità di visualizzare la catena di chiamate che porta ad un determinato metodo.

Biblioteca

ON LINE

Codetoad.com

Molti esempi in codice, tutorial, ed una miriade di altre risorse per tutti gli sviluppatori ASP, ASP.NET, HTML, JAVA, Visual Basic, XML

http://www.codetoad.com

DevASP.net

Per dirla così come si autosponsorizza il sito: il primo posto dove reperire informazioni su: ASP.NET, VB.NET, VS.NET, XML, SQL e C-Sharp.
Articoli, tutorial ed esempi in codice, sviluppati da veri esperti del mondo .NET

http://www.devasp.net

Experts Exchange

Un vero e proprio team d'esperti della programmazione che si ripromettono di aiutare lo sviluppatore nel risolvere i problemi più tediosi.
Il sito organizzato per categorie, consente di individuare facilmente l'area di proprio interesse.

http://www.experts-exchange.com/

Mathematics For Computer Graphics Applications Second Edition

Questo volume, ormai giunto alla seconda versione, figura, di fatto, come uno dei migliori e più autorevoli testi sulla Computer Graphics, una vera Bibbia adottata finanche in campo universitario.
Rappresenta il testo ideale per l'appassionato ed il professionista che vuole carpire tutti i concetti matematici che stanno alla base delle più avanzate tecniche di computer grafica.
Mortenson, l'autore del testo, nel suo volume affronta sia i concetti fondamentali (vettori, matrici, trasformazioni) che gli aspetti più avanzati (CAD/CAM, geometria solida, simmetria, curve di Bezier, superfici).

Difficoltà: Alta Autore: Michael E. Mortenson • Editore: Industrial Press
http://www.industrialpress.com **• ISBN: 0-8311-3111-X • Anno di pubblicazione: 1999**
Lingua: Inglese • Pagine: 416 • Prezzo: USD. 42,95

ASP.NET – La guida Completa 3/ed

ASP.NET – La guida Completa rappresenta lo strumento ideale per velocizzare il proprio lavoro con ASP .NET, la piattaforma di sviluppo Web presente nel nuovo Framework .NET di Microsoft.
Il volume affronta tematiche rivolte sia ai neofiti della programmazione, sia a chi senti un po' più esperto.
Tra gli argomenti trattati: servizi Web XML, database relazionali, tracing, registrazione e gestione degli eventi, la programmazione basata sui componenti e la gestione dei controlli personalizzati. Fornisce inoltre alcune informazioni utili sulla sicurezza per la salvaguardia del proprio sito e sull'iscrizione ai servizi.

Difficoltà: Medio-Alta • Autore: Matthew MacDonald • Editore: McGraw-Hill
http://www.mcgraw-hill.it **• ISBN: 88-386-4277-X • Anno di pubblicazione: 2002**
Lingua: Italiano • Pagine: 880 • Prezzo: € 54,00

VISUAL BASIC.NET – La guida Completa 3/ed

Il volume fornisce esaurienti informazioni sul nuovo linguaggio Visual Basic .NET; in un solo volume tutto quello che serve per mettere a frutto le innovative funzionalità del linguaggio: grammatica, flusso di controllo, operatori, classi e interfacce, strutture e raccolte di dati, componenti GUI, threading e debugging. Un immancabile strumento di riferimento per tutti coloro che voglio padroneggiare il nuovo linguaggio Microsoft.
Tra gli argomenti trattati:

- Il Common Language Runtime e l'ambiente d'esecuzione gestito dal Framework .NET;
- Creazione di tipi di valore ed enumeratori;
- Progettazione e implementazione di gerarchie di classi con l'ereditarietà;
- Utilizzo delle interfacce per algoritmi sofisticati.

Difficoltà: Medio-Alta • Autore: Jeffrey R. Shapiro • Editore: McGraw-Hill
http://www.mcgraw-hill.it **• ISBN: 88-386-4276-1 • Anno di pubblicazione: 2002**
Lingua: Italiano • Pagine: 856 • Prezzo: € 52,00

Sorting
metodi diretti

Gli algoritmi di ordinamento rivestono una cruciale rilevanza nell'ambito della programmazione, essi infatti, sono il presupposto dei più efficienti metodi di ricerca. La letteratura sul sorting è molto vasta, nel primo appuntamento sull'argomento analizziamo i metodi conosciuti come "diretti".

Che mantenere oggetti e informazioni in modo ordinato sia un'attività che faciliti la vita di tutti i giorni, soprattutto nel momento in cui si debbano ricercare oggetti e informazioni, ve lo posso garantire personalmente, a nome di tutti i disordinati del paese. Cari lettori, non descrivo la mia scrivania per pudore, ad ogni modo vi assicuro che mi crea qualche problema l'attività di ricerca di un qualsivoglia appunto, dispensa o documento. Certo quando si cerca il tempo è sempre pochissimo, come nei film in cui si deve disinnescare una bomba ad orologeria ed il conto alla rovescia inesorabilmente avanza, ed il protagonista pronuncia solitamente la frase "il tempo è maledettamente poco", ma all'ultimo momento ce la caviamo! Ormai mi sono assuefatto alla cosa e in questo mare magnum mi riesco a districare quasi sempre (me la cavo!) anche se con qualche problema, come dicevo. Noi disordinati potremmo dormire sonni tranquilli se soltanto mantenessimo le nostre cose in rigoroso ordine, tutto funzionerebbe a meraviglia e non ci sarebbe alcun rischio di esplosione, ma in tal caso non saremmo disordinati. E pensare che uno studio pedagogico ha dimostrato che una delle prime attività che i bambini apprendono è quella di ordinare, molto prima che si sviluppino molte altre capacità, come ad esempio quelle aritmetiche. Non a caso qualsiasi tipo di catalogo o elenco si presenta in modo ordinato. Pensate per un attimo, soltanto per un attimo, cosa significherebbe cercare un numero di telefono in un elenco non ordinato… beh, forse meglio non pensarci. Per concludere la premessa vi racconto cosa mi è accaduto qualche giorno fa: stavo cercando una rivista (non di informatica), sulla scrivania appena accennata, ma nello scartabellare ho incrociato una vecchia dispensa sul sorting, ecco perché ci tocca questo argomento. Ma non vi preoccupate, come sempre ci divertiremo!

PERCHÉ ORDINARE?

Dopo aver intuito quale può essere l'utilità di mantenere ordine nella quotidianità, cerchiamo di capire come la stessa attività ci possa aiutare nella nostra veste di programmatori di computer. Nell'elaborazione dati, come noto, ordinare è un processo di primaria importanza e lo scopo finale è sempre quello di agevolare la ricerca. In una lista ordinata è possibile effettuare la ricerca binaria e quindi evitare di adottare una scansione sequenziale in modo da abbattere la complessità computazionale da n a $log(n)$, con n ovviamente la lunghezza della lista. La differenza tra le due ricerche si fa sentire nel caso in cui il numero di elementi è elevato, si pensi all'elenco telefonico di una grande città, e il tempo di accesso ai dati o comunque di operazione elementari è significativo, come accade quando si trattano dati su file. Ma facciamo un passo indietro. Come d'abitudine esaminiamo la questione da un punto di vista teorico. Se abbiamo n elementi $a1, a2, \ldots, an$, essi saranno ordinati rispetto ad un criterio di confronto (ordine alfabetico se si tratta di parole, o nel caso numerico rispetto all'ordine intrinseco dell'insieme dei numeri interi) semplicemente se $a1 < a2 < \ldots < an$. In tal caso si parlerà di ordine crescente, mentre sarà decrescente se la relazione tra i vari elementi è di maggioranza. Ordinare significa quindi, considerare questo insieme di oggetti e operare su di essi una permutazione che verifichi la proprietà appena descritta. Nella vasta letteratura sull'argomento si distinguono molte tipologie di ordinamento; in questo appuntamento studieremo i metodi diretti caratterizzati da una complessità $O(n^2)$ quindi non molto efficienti se si considera che altri metodi (che tratteremo la prossima volta) hanno complessità di $O(n*log(n))$. I metodi diretti sono per così dire didattici, poiché esplicitano in algoritmo quello che senza ausilio di computer noi facciamo per ordinare degli oggetti (o almeno molti di noi fanno). Il codice che ne scaturisce è relativamente breve e di facile comprensione.

Altri metodi presentano un codice molto più lungo e complesso che, come spesso accade in programmazione, da risultati di migliore efficienza. Insomma è consigliabile esaminare i metodi diretti per comprendere a fondo le problematiche legate al sorting e per

Ordinamento interno ed esterno

Si parla di ordinamento interno quando la struttura dati su cui effettuare il sorting risiede su memoria "interna", ossia primaria. L'ordinamento esterno è applicato a file. Il differente tipo di accesso alle due strutture dati implica lo sviluppo di algoritmi diversi.

Soluzioni

Ricerca binaria e sequenziale

Data una lista di elementi, la ricerca sequenziale prevede la scansione dei valori in ordine di posto a partire dal primo fin quando non si trova l'elemento desiderato. Nel caso peggiore bisogna effettuare *n* confronti mentre in quello medio *n/2*. Ad ogni modo si parla di complessità computazionale di ordine *n*. La ricerca binaria è applicabile soltanto a liste ordinate ed è un metodo che, ad ogni fase, prevede la valutazione di metà dell'insieme precedentemente esaminato.
Nel caso di un vettore si confronta l'elemento da ricercare con quello mediano del vettore; se è stato trovato, la ricerca è terminata. Altrimenti, se l'elemento da ricercare è maggiore di quello del vettore, si scarta la prima metà dell'array e si focalizza l'interesse sull'altra metà, nel caso contrario ad essere eliminata sarà la seconda meta. Nella parte di vettore focalizzato si ripeterà il procedimento. Il processo sarà iterato fin quando non si trova l'elemento. Questo algoritmo garantisce complessità dell'ordine *log (n)*, quindi molto più veloce.

avere un giusta preparazione per affrontare i metodi avanzati. Un'altra ipotesi che si fa nello sviluppo di tali algoritmi è di evitare lo spreco di memoria. Sebbene oggi, i moderni elaboratori dispongano di memorie molto "capienti" è pur vero che spesso si ha a che fare con quantità di dati davvero "enormi", ciò significa che non si duplicheranno vettori ma si farà riferimento all'incirca ad una quantità di memoria sufficiente a contenere i dati. Del resto, se così non fosse, gli algoritmi di sorting perderebbero anche di interesse, e non sarebbe possibile sviluppare una vera e propria teoria dell'ordinamento come invece esiste. Inoltre, vedremo che i vari metodi sono fortemente dipendenti dalla struttura dati usata (anche nell'ambito di dati nello stesso tipo di memoria, come ad esempio: vettori e liste a puntatori) e dalla tipologia di dati (ad esempio se i dati sono quasi del tutto ordinati converrà usare un metodo piuttosto che un altro), ma man mano che affronteremo i metodi faremo riferimento a questi aspetti. Adesso preoccupiamoci di esaminare tre diversi tipi di algoritmi di sorting.

I PRIMI PASSI

Prima di sviluppare i vari algoritmi è necessario definire la struttura dati su cui operare il sorting. Il caso più generale prevede un vettore di record, dove ogni record consta di una serie di campi, di cui uno *chiave* per operare i confronti che producono l'ordinamento. Per snellire il codice, che svilupperemo in C++, (uno dei nostri linguaggi preferiti, per le naturali caratteristiche didattiche che detiene), considereremo un semplice vettore di interi.

```
// Sortbas.cpp
#include<iostream.h>
#include<conio.h>
int *a,*b, n;
enum bool { false, true };
void inser (int *);
void vis (int *);
bool is_sort (int *);
void selectionsort (int *)
void bubblesort(int *)
void bubblesortev(int *)
void insertionsort (int *)
void insertionsortev(int *)
```

Data la relativa semplicità e la non eccessiva occupazione di spazio ho preferito riportare per intero il codice prodotto e compilato in C++. Le prime linee di codice mostrano, oltre che l'intestazione in cui vengono richiamate le librerie per la gestione dei flussi di I/O e per alcune ruotine di gestione video, le variabili in gioco. Sono stati dichiarati due vettori di interi di nome a e b, preciso a chi non ha confidenza con C++ che in questo linguaggio non ci sono differenze tra puntatori e vettori, per tale ragione e poiché è più naturale la gestione dei parametri, ho dichiarato i due vettori come puntatori. La variabile globale *n* è la dimensione del vettore. Mentre *bool* è il tipo binario (in Pascal è conosciuto come *boolean*). Ancora di seguito sono riportati i prototipi di tutte le funzioni implementate in seguito. Le due funzioni *inser(int *)* e *vis(int *)* si occupano rispettivamente di inserire i dati nel vettore e di visualizzarli a video, entrambe hanno come parametro in ingresso l'array. Mentre *is_sort (int *)* è una funzione booleana che verifica se il vettore è ordinato secondo la proprietà esposta nel paragrafo precedente.
Ecco le implementazioni di queste routine.

```
void inser (int *arr)
{  int i;
        cout<<"inserisci il numero di elementi :";
        cin>>n;
        for (i=1; i<=n; i++)
        {   cout<<i<<" -- > ";
            cin>>arr[i]; }
}
void vis (int *arr)
{  int i;
        cout<<"Ecco il vettore \n";
        for (i=1; i<=n; i++)
        {   cout<<arr[i]<<"\n ";
        };
        getch(); //blocca sullo schermo l'output }
bool is_sort(int *arr)
{  bool sort=true;
        int i;
        for (i=1; i<=n-1; i++)
        { if (arr[i]>arr[i+1]) sort = false; };
        return sort;
}
```

SELECTION SORT

Uno dei più immediati ed intuitivi metodi di ordinamento è quello per selezione. L'ordinamento avviene attraverso *n* passate (o fasi), dove ognuna di esse consta di una scansione del vettore, da qui la complessità $n*n$ ossia n^2. Ad ogni passata si individua l'elemento più piccolo ed il suo indice nella porzione di vettore presa in considerazione. In particolare, al primo passaggio si tiene in considerazione l'intero array e una volta individuato il minimo viene posto al primo elemento. La seconda passata ripete la stessa operazione sulla porzione di vettore ottenuta dalla esclusione del primo elemento che nella prima fase è stato collocato nella posizione corretta.

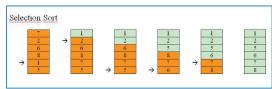



Fig. 1: **La freccia indica il valore minore. Le zone in verde sono ordinate.**

Soluzioni

Per intenderci dopo *m* passaggi (ovviamente valore minore di *n*) risultano ordinati i primi *m* numeri del vettore mentre la restante porzione di array di *n-m* elementi deve essere ancora ordinata. In Fig. 1 vengono descritte in forma grafica le varie fasi di ordinamento di un vettore con il metodo di selezione.

```
void selectionsort (int *arr)
{   int i,j,min,indmin;
for (i=1; i<=n-1; i++)
            {   indmin=i;   min=arr[i];
                for (j=i+1; j<=n; j++)
                        if (arr[j]<min) {   indmin=j;
                                              min=arr[j];};
                    arr[indmin]=a[i]; arr[i]=min; }
}
```

Il ciclo esterno associato alla variabile *i* effettua le varie passate mentre quello interno associato alla variabile *j* si occupa di trovare il minimo *(min)* e il corrispondente indice *(indmin)* della porzione di vettore che si tiene in considerazione. Una variante del metodo, anch'essa diffusa per il carattere didattico, ma non molto efficiente, prevede di effettuare un maggior numero di confronti e scambi. In particolare non viene calcolato il valore minimo e il corrispettivo indice ma ogni qual volta si esamina un generico elemento, lo si mette in relazione con il primo parziale della lista e, se risulta non ordinato, avviene lo scambio. Anche in questo caso si parla di parziale in virtù del fatto che al primo passo si considera l'intero vettore, al secondo il vettore tranne il primo elemento e così via.

BUBBLE SORT

In questo tipo di ordinamento sono previsti molti scambi, in particolare vengono confrontati sempre elementi adiacenti; infatti, in alcuni testi lo si trova etichettato come ordinamento per scambio diretto. Per realizzarlo si utilizzano come nel caso precedente due cicli innestati che sanciscono nuovamente, come tutti i metodi diretti del resto, una complessità O(n2). Il ciclo interno su j permette il confronto di tutte le coppie adiacenti di elementi e qualora non siano ordinati vengono scambiati (la variabile comodo consente un corretto scambio). Al termine di ogni passata viene depositato il valore più grande in fondo, come se una bolla passasse nella struttura portando con se il valore maggiore da lasciare all'ultimo posto, da qui ha origine il nome del metodo. Ovviamente, nella prima fase il vettore è considerato per intero, nella seconda è tutto tranne l'ultimo elemento (già collocato) e cosi via. Si consulti la figura 2 per puntualizzare il metodo.

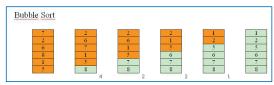



Fig. 2: **Le zone in verde sono ordinate. I numeri in basso indicano il numero di scambi per quella passata.**

```
void bubblesort(int *arr)
{   int i,j,comodo;
    for (i=1; i<=n-1; i++)
    for (j=1; j<=n-i; j++)
    if (arr[j]>arr[j+1]) {  comodo=arr[j];
        arr[j]=arr[j+1];
        arr[j+1]=comodo; };
}
```

Può accadere che il vettore raggiunga la permutazione ordinata prima che si esauriscano tutte le fasi. In tal caso queste ultime vengono fatte a vuoto con evidente perdita di tempo. Da notare che la situazione non è affatto rara. Quindi, un netto miglioramento del metodo si ottiene evitando di eseguire i cicli a vuoto. Nella versione evoluta della funzione viene introdotta una variabile booleana scambio per segnalare eventuali scambi. Qualora non si effettuano scambi allora vuol dire che il vettore è ordinato.

```
void bubblesortev(int *arr)
{   int i,j,comodo;
    bool scambio=false;
    for (i=1; ((i<=n-1) || (!scambio)); i++)
    {   scambio=false;
        for (j=1; j<=n-i; j++)
        if (arr[j]>arr[j+1]) {  scambio=true;
            comodo=arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=comodo; }
    }
}
```

Ricordo che in C++ l'istruzione for prevede che si possa uscire dal ciclo anche al verificarsi di una condizione composta, come in questo caso, in cui è presente un *or* (doppia sbarretta | |). Il punto esclamativo è il connettivo logico *not*. Un ulteriore sviluppo del bubble sort è conosciuto come *shake sort*, in questo metodo si tiene conto dell'indice dell'ultimo scambio, evitando così di considerare porzioni di vettori già ordinate, inoltre ad ogni passata si cambia verso alla bolla (*alto-basso* è alternato con *basso-alto*), da qui il nome *shake*. Questa che sembra una banalità è invece molto utile, se ad esempio abbiamo un valore molto piccolo in fondo al vettore con il bubblesort tradizionale si rendono necessari molti cicli per riportarlo su (ogni passata conquista una posizione più bassa) mentre se si inverte il senso della bolla, questo elemento viene riportato su con un solo passaggio. Utile nelle tipologie di dati in cui il vettore è "quasi" ordinato: tipici casi riconducibili a tali tipologie si hanno quando ad un vettore ordinato vengono effettuate delle *append* (aggiunti pochi elementi). In

Ordinamento stabile

La caratteristica di stabilità di un metodo di sorting è verificata se, per valori identici della chiave, si mantiene inalterato l'ordinamento originario degli elementi. Se abbiamo a che fare con un vettore di record in cui bisogna ordinare per il campo chiave, può capitare che alcuni elementi dell'array presentino la stessa chiave, in tal caso si dice che l'ordinamento è *stabile* se alla fine del processo di sorting per questi elementi è stato mantenuto l'ordine iniziale (o relativo se effettato rispetto ad una chiave secondaria, ossia su un secondo campo del record).

Permutazioni

Se consideriamo *n* oggetti, permutare significa cambiare di posto uno o più oggetti. Come il calcolo combinatorio insegna, si parla di permutazioni semplici nel caso particolare di disposizioni semplici di n elementi della classe *n* (ossia disposti a gruppi di *n*). Tutte le possibili permutazioni di *n* elementi sono il fattoriale di n:

```
Pn=n!
=n*(n-1)*(n-2)*..*1
P0=1
```

Se nella lista vi sono elementi ripetuti il numero di permutazioni è minore.

tal caso è fortemente consigliato lo shakesort.

INSERTION SORT

La rassegna degli algoritmi di sorting diretti si conclude con quello per inserimento. Il metodo in questione prevede di esaminare un vettore di grandezza via via crescente. Al primo passo è di lunghezza *uno* (si considera quindi solo il primo elemento) e bisogna collocare il secondo elemento al posto giusto. Al secondo i primi due elementi risultano in ordine è tocca al terzo essere posto correttamente. Iterando il procedimento si perviene all'ordinamento completo. I passaggi che portano alla risoluzione sono rappresentati in Fig. 3.

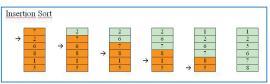



Fig. 3: **Le zone in verde sono ordinate. La freccia indica l'elemento da inserire.**

Osservando il codice sviluppato si può notare come il ciclo interno questa volta faccia scorrere l'indice nel senso contrario, e come la condizione di uscita dallo stesso sia l'assenza di ordine tra l'elemento da collocare *elem* e il valore corrente del vettore. Intanto che non viene individuato il posto dove collocare *elem*, ad ogni generica passata avviene la traslazione della porzione di vettore. In posizione *zero* (usata per comodità visto che i valori effettivi occupano le posizioni da 1 a n) del vettore assegniamo l'elemento da inserire per avere la certezza che almeno l'ultimo confronto risulti falso e consentendo quindi di assegnare almeno in prima posizione (nel caso in cui si tratta dell'elemento più piccolo) il valore in questione.

```
void insertionsortev(int *arr)
{   int i,j,elem;
        for (i=2; i<=n; i++)
        {   elem=arr[i];
            arr[0]=elem;
            for (j=i-1; elem<arr[j]; j--)
                arr[j+1]=arr[j];
          arr[j+1]=elem;
        }
}
```

Il metodo appena esaminato si adatta bene alla struttura dati lista a puntatori, poiché per tale struttura non bisogna effettuare alcuna traslazione, è sufficiente scorrere sui vari nodi ed al momento opportuno inserire il nuovo elemento predisponendo i link del precedente e successivo nodo per compiere l'*insertion*. Anche in questo caso si può formulare un utile miglioramento che aumenta l'efficienza del metodo. La ricerca del punto in cui inserire l'elemento corrente può essere attuata dicotomicamente per abbattere i tempi di collocazione.

L'evoluzione modifica l'algoritmo precedente nella fase di ricerca, effettuata appunto con il metodo dicotomico. Il ciclo di *while* si occupa di tale fase. Il *for* successivo trasla tutto l'array di un elemento.

```
void insertionsort (int *arr)
{   int i,j,elem,iniz,fine,pos;
        for (i=2; i<=n; i++)
        {   elem=arr[i];
      iniz=1; fine=i-1;
            while (iniz<=fine)
            {  pos=(iniz+fine)/2 ;
                  if (elem<arr[pos]) fine=pos-1;
                  else iniz=pos+1;
            }
            for (j=i-1; j<=iniz ; j--)
                 arr[j+1]=arr[j];
                 arr[iniz]=elem;
        }
}
```

CONCLUSIONI

Per completezza, e a titolo di esempio, presentiamo un possibile *main program* che richiami le funzioni sviluppate. Nell'esempio le funzioni sono associate a due differenti vettori. Gli ordinamenti richiamati sono a inserimento. Un utile miglioramento prevede un menu di accesso alle varie routine.

```
int main()
{ /*    clrscr();
        inser(a);
        vis(a);
        insertionsort(a);
        if (is_sort(a)) cout<<"ordinato"<<"\n";
        else cout<<"disordinato\n";
        vis(a); */
        clrscr();
        inser(b);
        vis(b);
        insertionsortev(b);
       // if (is_sort(b)) cout<<"ordinato"<<"\n";
       // else cout<<"disordinato\n";
        vis(b); }
```

Con il primo articolo sul sorting abbiamo introdotto l'argomento esaminando le comuni problematiche e tentando una prima classificazione. Abbiamo, inoltre, analizzato la prima tipologia di algoritmi, quelli diretti. La prossima volta saranno presenteti algoritmi avanzati. A proposito, dimenticavo! Volevo precisare che nel ruolo di programmatore sono ordinato, come gli affezionati lettori possono confermare, in questo ambito mi viene naturale, è con la scrivania che non riesco ad applicare alcun valido algoritmo.

Tenterò ancora qualche metodo e vi farò sapere!

Fabio Grimaldi

Algoritmi

GENETICI IN DELPHI 6 UN' APPLICAZIONE PRATICA

Gli Algoritmi Genetici, nati negli anni sessanta, sono oggetto di sempre maggiori attenzioni da parte degli sviluppatori impegnati in ottimizzazione di processi.
La struttura degli Algoritmi Genetici ha alcune caratteristiche che li accomuna in maniera sorprendente con i meccanismi evolutivi degli organismi viventi.
Vediamo come applicare queste tecniche sorprendenti, fornendo un'applicazione pratica a quanto esposto teoricamente, nel numero 60 di ioProgrammo.

La ricerca di un componente o di un motore logico adatto alla risoluzione di un problema generico, è sempre stato il sogno di chi scrive algoritmi di ottimizzazione e di *'Problem Solving'*. Di recente si assiste con una sempre maggiore attenzione allo sviluppo dell'"Evolutionary Computation" che sta promettendo grandi novità nel campo dell'ottimizzazione dei processi e della risoluzione dei problemi. Queste metodologie di programmazione si basano sull'evoluzione di una popolazione di individui che rappresentano inizialmente ciascuno una possibile soluzione, operando poi una opportuna selezione e riproduzione si ottiene il risultato finale. Non si scenderà nel dettaglio teorico di quanto il lettore potrà trovare nell'articolo dell'autore *'Introduzione all'Evolutionary Computation'* pubblicato nel numero 60 di questa rivista per maggiori approfondimenti.

ALGORITMI GENETICI: LA RICERCA DEI MASSIMI DI UNA FUNZIONE

Leggendo vari testi che trattano a livello teorico il campo degli Algoritmi Genetici, talvolta si ha la sensazione di affrontare problematiche astratte e difficilmente applicabili ad un problema pratico. In queste pagine vogliamo raggiungere il duplice obiettivo di dimostrare che queste tecniche sono in effetti utili alla risoluzione di uno svariato numero di problemi pratici e che allo stesso tempo è possibile sviluppare una applicazione che sia in grado di risolvere una astrazione teorica del problema, codificata sotto forma di funzione matematica. Se da un lato quanto appena affermato farebbe brillare gli occhi del mio ormai anziano professore di Analisi Matematica, dall'altro comporta uno sforzo concettuale non comune. L'aspetto sorprendente di quanto andremo ad analizzare è che l'applicazione ed il relativo componente Delphi che andremo a sviluppare, si comportano come un motore di 'Problem Solving' che è in grado di risolvere la ricerca dei massimi di una funzione in modo INDIPENDENTE dalla funzione stessa. Si badi bene che la ricerca del massimo non avviene per 'scansione' di tutti i valori di $f(x)$ per un dato intervallo $(X0,X1)$, mediante la tecnica comunemente conosciuta come *'Hillclimbing'*, ma seguendo una ricerca 'Parallela' generando contemporaneamente una popolazione di individui (Valori codificati di X) che hanno un proprio valore di 'Fitness' (corrispondente alla funzione $f(x)$) nell'intervallo in esame. Ciascun individuo sarà rappresentato da una matrice che raffigura la schematizzazione di un cromosoma comprendente le caratteristiche di quel particolare membro della popolazione. Si ricerca così un parallelismo con i metodi che regolano la genetica e quindi la vita negli organismi biologici. In analogia con il mondo naturale, infatti, anche gli Algoritmi Genetici hanno alcune caratteristiche che li accomunano in maniera sorprendente con l'evoluzione degli organismi viventi. Il programma dimostra come possa essere possibile individuare i massimi di una funzione, definita in un dato intervallo, utilizzando il componente Delphi *'GeneticAlgComponent1_0'*, capace di risolvere una qualsiasi funzione di una variabile. Il programma è dotato di quattro pulsanti, che a titolo di esempio 'lanciano' l'esecuzione dei discendenti del nostro com-

File sul CD

Il componente *'GeneticAlgComponent1_0'* insieme al codice ed all'applicazione completa, compilata e funzionante è compreso nel CD allegato alla rivista nel file:

Algorithm_1_0.zip

Algoritmi Genetici

Algoritmi
Genetici in Delphi 6
Un' applicazione pratica

ponente sui quali è stato effettuato l'override della funzione da studiare contenuta in *'ObjFunc'*, come verrà descritto più nel dettaglio di seguito. Si può effettuare il 'Docking' dei componenti semplicemente trascinandone il bordo, per rendere possibile l'analisi contemporanea dello studio di più funzioni, come riportato in Fig. 1.

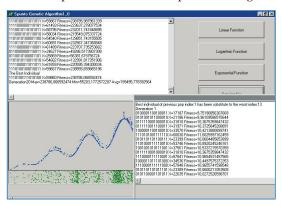



Fig. 1: Il programma proposto in queste pagine è in grado di determinare i massimi di una funzione utilizzando le tecniche di programmazione conosciute come 'Algoritmi Genetici'.

IL COMPONENTE DELPHI 'GENETICALGCOMPONENT 1_0'

Il componente *'GeneticAlgComponent1_0'* riportato di seguito è compreso nel CD allegato alla rivista nel file *'Algorithm_1_0.zip'* insieme al codice e all'applicazione completa compilata e funzionante. La prima particolarità di questo componente è che discende direttamente da *'Tpanel'* dal quale ovviamente eredita tutte le caratteristiche: in aggiunta a ciò ingloba al suo interno anche un componente *Tmemo*, utilizzato per visualizzare le operazioni ed i valori numerici della popolazione in esame, oltre ad una Paintbox, utile per la rappresentazione grafica della funzione e degli individui della popolazione. Analizziamo ora brevemente le funzioni e le procedure cardine del componente:

```
unit GeneticAlgComponent1_0;
TSpuntoGA = class(TPanel)
 private
  { Private declarations }
   Panel:TPanel;
  { Public declarations }
    procedure PaintLine(Canvas: TCanvas; I, Len:Integer);
    procedure PaintEllipse(Canvas: TCanvas;
                              X,Y,Radius: Integer; color:Tcolor);
    procedure DrawPaintBox(Box: TPaintBox);
    procedure pause(pauselength:integer);
    procedure page(var out:text);
```

Le procedure riportate fino a questo punto, contenute nella parte *'Private'* della classe, hanno lo scopo di facilitare la rappresentazione grafica della funzione ed il loro scopo è abbastanza ovvio, possiamo notare *Paintline* e *Paintellipse* che si occupano di disegnare linee ed ellissi.

```
procedure repchar(var out:text; ch:char;
                                    repcount:integer);
   procedure skip(var out:text; skipcount:integer);
   procedure initialize;
   procedure writechrom(var outstring:String;
                  chrom:chromosome; lchrom:integer);
   procedure report(gen:integer);
```

In questo blocco di codice si hanno alcune procedure che servono alla gestione interna della nostra popolazione di individui.

```
   procedure ManageBestIndividual;
   function select;
   function mutation;
   procedure crossover;
   procedure generation;
```

Finalmente giungiamo alla definizione delle funzioni e procedure che gestiscono in modo diretto la selezione, mutazione e crossover della popolazione, oltre ad una procedura molto importante che tratta l'archiviazione del migliore individuo, quando la proprietà *'GAKeepBestIndividual'* è impostata come *'True'*. Per maggiori dettagli su questa parte del codice, si rimanda il lettore per motivi di spazio, all'articolo *'Introduzione all'Evolutionary Computation'* pubblicato nel numero 60 di questa rivista.

```
 protected
  { Protected declarations }
 public
   Memo:Tmemo;
   PaintBox:Tpaintbox;
   constructor Create(AOwner: TComponent); Override;
   function objfunc(x:real):real; Virtual;
   function decode(chrom:chromosome;
                              lbits:integer):real; Virtual;
```

Oltre alla presenza, come abbiamo accennato di una *'Memo'* ed una *'Paintbox'*, notiamo le procedure *'Objfunc'* che contiene la funzione oggetto da studiare (per default la funzione nativa nel componente è $f(x)= x\ sen(x)$) e *'Decode'*, che decodifica il valore reale della stinga di bit rappresentanti il cromosoma *'crom'*: *'lbits'* rappresenta il numero di bit del cromosoma stesso.

```
   procedure statistics(popsize:integer;var
                        max,avg,min,sumfitness:real;
      var pop:population; Var
                         LocalBest,LocalWorst:individual;
      var Lbestindex,LworstIndex:integer);
```

Installazione del componente

Per installare il componente, è sufficiente eseguire il comando *'Install Component'* dal menù *'Component'*, selezionare la Tag *'Into new Package'* ed inserire nella linea *'Package file name'*: *SpuntoGeneticAlgoritm.dpk*, od un nome equivalente, oltre, ovviamente a selezionare il componente *'GeneticAlgComponent1_0.dcu'* nella linea *'Unit file name'*.

```
procedure MainGO;
Function  GetMaxPop:Integer;
Procedure SetMaxPop(I:Integer);
Function  Getmaxstring:Integer;
Procedure Setmaxstring(I:Integer);
```

Al termine della parte *Public* si hanno altre procedure inerenti al riporto statistico della popolazione. La procedura *'MainGO'* lancia l'esecuzione dell'algoritmo genetico.

```
published
  { Published declarations }
  Property GANmaxpop:Integer Read GetMaxPop Write
                                        SetMaxPop;
  Property GANmaxstring:Integer Read Getmaxstring
                                   Write Setmaxstring;
  Property GAIndividualMax:Real Read Max Write Max;
  Property GAIndividualMin:Real Read Min Write Min;
  Property GAIndividualAvg:Real Read Avg Write Avg;
  Property GAPopulationSize:Integer Read PopSize
                                        Write PopSize;
  Property GAIndChromLenght:Integer Read LChrom
                                        Write LChrom;
  Property GAGenerationNumber:Integer Read MaxGen
                                        Write MaxGen;
  Property GAProbCrossover:Real Read PCross Write
                                          PCross;
  Property GAProbMutation:Real Read PMutation Write
                                       PMutation;
  Property GAKeepBestIndividual:Boolean Read
         KeepBestIndividual Write KeepBestIndividual;
end;
```

Nella parte *'Published'* abbiamo tutte le proprietà che vengono visualizzate nell'*Object Inspector*, una volta rilasciato il componente a *'Design Time'* sull'applicazione, ovviamente prima di fare ciò è necessario installare il componente. Le proprietà relative al nostro componente per la risoluzione di algoritmi genetici, iniziano tutte con la coppia di lettere *'GA'* per renderne più facile l'individuazione attraverso l'Object Inspector.
Oltre alle proprietà importanti ai fini statistici come *GAIndividualMax*, *GAIndividualMin* e *GAIndividualAvg*, che restituiscono massimo, minimo e media del valore di *'Fitness'* della popolazione, si evidenziano per importanza *GAProbCrossover*, *GAProbMutation*, *GAIndChromLenght* e *GAPopulationSize* che individuano rispettivamente le probabilità di Crossover, mutazione ed i parametri fisici della popolazione, cioè la lunghezza del cromosoma di ciascun individuo ed il numero di individui della popolazione.

IL PROGRAMMA DI STUDIO DI UNA FUNZIONE

L'utilizzo approfondito del componente appena descritto nell'essenzialità della propria struttura esula dallo scopo dimostrativo di questa sede, tuttavia si può dire che variando i valori di *GAProbCrossover*, *GAProbMutation*, *GAIndChromLenght* e *GAPopulationSize* si varia in modo considerevole il comportamento dell'algoritmo, in termini di velocità di convergenza verso il massimo e capacità di individuare più massimi: l'esplorazione di queste caratteristiche avanzate è stata oggetto di studio in svariati documenti reperibili attraverso i link riportati a lato.

```
type
 TLinearGA = Class(TSpuntoGA)
   Function ObjFunc(x:real):real; Override;
 End;
 TExponentialGA = Class(TSpuntoGA)
   Function ObjFunc(x:real):real; Override;
 End;
 TLogaritmicGA = Class(TSpuntoGA)
   Function ObjFunc(x:real):real; Override;
 End;
```

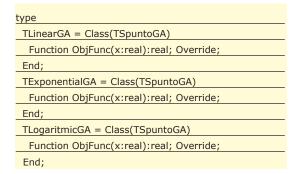
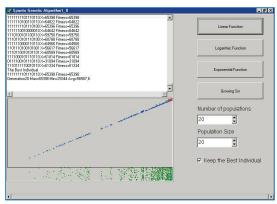



Fig. 2: **La figura mostra l'esecuzione dell'algoritmo per la ricerca dei massimi della funzione f(x)=x; si noti il valore numerico del massimo e la sua rappresentazione grafica in rosso.**

Vogliamo analizzare quattro funzioni di esempio: una retta, un logaritmo, un esponenziale ed una funzione oscillatoria crescente. Per fare ciò definiamo i discendenti di *TspuntoGA* ed effettuiamo l'override della funzione *ObjFunc* che contiene appunto la funzione in analisi: facendo ciò non abbiamo minimamente modificato le caratteristiche operative del nostro 'Motore Genetico', gli abbiamo soltanto passato tre nuove funzioni da analizzare (la quarta è contenuta nella classe originaria).

```
TGeneticAlgorithm1_0Application = class(TForm)
  LinearFunction: TBitBtn;
  ApplicationPanel: TPanel;
  LogaritmicFunction: TBitBtn;
  ExponentialFunction: TBitBtn;
  GrowingSin: TBitBtn;
  NumberofPopolations: TSpinEdit;
  Label1: TLabel;
```

Algoritmi Genetici

Algoritmi
Genetici in Delphi 6
Un' applicazione pratica

Sul Web

http://www-illigal.ge.uiuc.edu/
GA Lab

http://www-doi.ge.uiuc.edu
The Design of Innovation

http://online.engr.uiuc.edu/shortcourses
TDOI Shortcourse

http://online.cen.uiuc.edu/webcourses/ge485/
GA Course

http://www-advancedgec.ge.uiuc.edu/
Advanced GEC Course

http://www.ge.uiuc.edu/tec/
Technology Entrepreneur Ctr

http://www-bplan.ge.uiuc.edu/
Business Plan Wrkshp Course

http://www.davidegoldberg.com/
GA/Innovation Consulting

http://www.wkap.nl/prod/s/GENA
Kluwer Series on GAs & EC

Algoritmi Genetici

Algoritmi
Genetici in Delphi 6 Un' applicazione pratica

```
  Label2: TLabel;
  PopulationSize: TSpinEdit;
  KeepBestIndividual: TCheckBox;
  procedure LinearFunctionClick(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure LogaritmicFunctionClick(Sender: TObject);
  procedure ExponentialFunctionClick(Sender: TObject);
  procedure GrowingSinClick(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
```

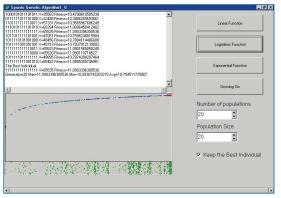



Fig. 3: L'immagine mostra la risoluzione di f(x) = log(x). Si noti la concentrazione degli indivdui delle popolazioni in prossimità del massimo (in verde).

L'applicazione ha una struttura molto semplice: possiamo notate i quattro pulsanti che servono a lanciare l'esecuzione della risoluzione delle quattro funzioni, uno *'Spinedit'* di nome *'PopulationSize'* che definisce il numero di individui componenti la popolazione che unito a *'NumberofPopolations'* stabilisce il criterio termine dell'algoritmo, individuato semplicemente da un numero massimo di generazioni.

```
var
  GeneticAlgorithm1_0Application:
TGeneticAlgorithm1_0Application;
  LinearGA:     TLinearGA;
  ExponentialGA:TExponentialGA;
  LogaritmicGA: TLogaritmicGA;
  GrowingSinGA: TSpuntoGA;
```

Nella parte riservata alle definizioni delle variabili del programma appaiono le istanze delle quattro classi che rappresentano i moduli di risoluzione delle quattro funzioni.

```
Function TLinearGA.ObjFunc(x:real):real;
Begin
  objfunc := X;
End;
Function TExponentialGA.ObjFunc(x:real):real;
Begin
  objfunc := Exp(X/10000);
End;
Function TLogaritmicGA.ObjFunc(x:real):real;
Begin
  objfunc := ln(X);
End;
```

Come è stato accennato in precedenza, oltre alla funzione inglobata nella classe principale *TspuntoGA*, che contiene la funzione $f(x)= x \sin(x)$, per analizzare anche le altre tre funzioni che ci siamo prefissati di studiare, ovvero una retta, un esponenziale ed un logaritmo, occorre riscrivere ed operare l'override di *'ObjFunc'*, contenente la funzione oggetto.

```
procedure TGeneticAlgorithm1_0Application
                    .LinearFunctionClick(Sender: TObject);
begin
LinearGa.Dock(self,rect(0,0,500,500));
LinearGA.Visible:=True;
LinearGA.GAGenerationNumber:=NumberofPopolations.
                                              Value;
LinearGA.GAPopulationSize:=PopulationSize.Value;
LinearGa.GAKeepBestIndividual:=
                          KeepBestIndividual.Checked;
LinearGA.MainGO;
end;
```

Ciascun pulsante presente sulla Form della nostra applicazione è dotato di un 'event handler' che ne gestisce la pressione: qui viene riportato il gestore dell'evento relativo alla funzione $f(x)=x$, corrispondente ovviamente ad una retta. Viene inizialmente stabilita la posizione e le dimensione di 'Docking' del componente, che viene inoltre reso visibile. Vengono inizializzati i parametri di numero massimo di generazioni *(GAGenerationNumber)* e di numero di individui componenti la popolazione *(GAPopulationSize)*, ricavandone i valori dai rispettivi spinedit. Dopo avere comunicato al componente se conservare l'individuo migliore di ciascuna popolazione (Nel caso sia *GAKeepBestIndividual=True*) viene lanciata l'esecuzione dell'algoritmo mediante la procedura *'MainGO'*. Gli *'Event handlers'* delle altre funzioni sono simili a quello appena descritto e possono essere consultati dal CD allegato.

Alla creazione dell'applicazione, viene lanciata la procedura *'GeneticAlgorithm1_0Application FormCreate'*, trovate il codice sul CD allegato, che si occupa di creare una istanza per ciascuna delle classi contenenti la funzione da studiare, assegnarne un parent e definirne dimensioni e font da utilizzare durante l'esecuzione del programma. Vengono inoltre stabilite le opzioni che permettono l'utilizzo del docking dei componenti, che non sono altro che *'Compound compo-*

Bibliografia

• **GENETIC ALGORITHMS IN SEARCH, OPTIMIZATION, AND MACHINE LEARNING**
David E. Goldberg

nents' creati a runtime e discendenti di Tpanel, per i quali è conveniente e possibile utilizzare tali tecniche.

ANALISI DELL'EFFICIENZA DELL'ALGORITMO

Abbiamo descritto finora il componente cardine e la sua implementazione in un'applicazione funzionante. Verifichiamo a questo punto se l'algoritmo soddisfa quanto ci siamo preposti di verificare all'inizio, ovvero che queste tecniche sono in effetti utili alla risoluzione di uno svariato numero di problemi pratici e che allo stesso tempo è possibile sviluppare una applicazione che sia in grado di risolvere una astrazione teorica del problema, codificata sotto forma di funzione matematica. Questa seconda affermazione viene presto dimostrata lanciando il programma e verificando che l'algoritmo individua il massimo di ogni funzione che gli viene passata, come si può verificare osservando le figure riportate in queste pagine relative allo studio delle varie funzioni. Il metodo di ricerca è veloce, e lo è tanto di più quanto più complessa è la funzione da studiare e quanto più alto è il numero di variabili, ovviamente facendo un paragone con i metodi di ricerca dei massimi e di ottimizzazione dei processi classici. Il fatto che la soluzione sia approssimata non è necessariamente una limitazione, a patto che l'approssimazione sia accettabile per l'applicazione che si intende sviluppare. Non sempre infatti abbiamo a disposizione un metodo o una formula che ci permetta il raggiungimento della soluzione esatta, oppure la risoluzione con questo tipo di approccio al problema può essere troppo complesso, o ancora il tempo di calcolo che si impiegherebbe potrebbe essere eccessivo. Infine possiamo affermare che l'applicazione di queste tecniche faciliterebbe la risoluzione di svariati problemi pratici, specialmente di ottimizzazione di processi, andando al di là dei soliti esempi accademici come il *'Traveling Salesman Problem'*, l'enigma delle regine oppure la ricerca di percorsi in un labirinto. Si pensi all'ottimizzazione di un processo di verniciatura, ad esempio, dove entrano in gioco decine di variabili, l'ottimizzazione del quale con metodi analitici o di *'Hillclimbing'* impiegherebbe milioni di anni di tempo di calcolo, potrebbe essere risolto in giorni, in modo certamente approssimato, ma sicuramente efficace. La definizione di percorsi per robot, il movimento dei bracci meccanici per l'industria assumerebbero una connotazione più 'intelligente' dando maggiore flessibilità alle apparecchiature. La funzione in questione, nonostante sia oscillatoria e dotata di più massimi parziali, viene risolta dall'algoritmo senza difficoltà.

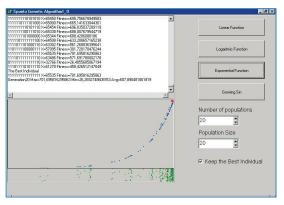



Fig. 4: L'immagine mostra la risoluzione di f(x) = exp(x).

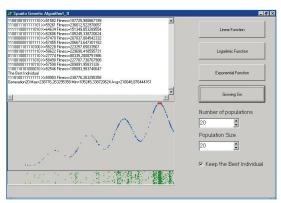



Fig. 5: L'immagine mostra la risoluzione di f(x) = x sin(x).

CONCLUSIONI

L'approccio alla risoluzione dei problemi attraverso gli algoritmi genetici è quantomeno affascinante, perché propone un metodo che è relativamente indipendente dalla natura del problema stesso. Abbiamo proposto in queste pagine un componente Delphi idoneo alla risoluzione di problemi di ricerca di massimi di una funzione e quindi elemento base per una applicazione di ottimizzazione di processi. E' stata sviluppata una applicazione completa per la verifica del componente che si è dimostrato in grado di risolvere qualsiasi funzione gli sia stata proposta. Abbiamo inoltre dimostrato che le tecniche basate sugli Algoritmi Genetici sono in effetti utili alla risoluzione di uno svariato numero di problemi pratici e che allo stesso tempo è possibile sviluppare una applicazione che sia in grado di risolvere una astrazione teorica di un problema, codificato sotto forma di funzione matematica. Ci si propone di sviluppare in futuro una applicazione di ottimizzazione di un processo complesso, regolato da più variabili, al fine di dimostrare le potenzialità di queste tecniche e di operare un reale salto di qualità nella gestione dei parametri di un qualunque processo industriale.

Luca Spuntoni

Contattare l'autore

L'autore è lieto di accettare qualsiasi consiglio e di rispondere a qualunque domanda dei lettori all'indirizzo di posta elettronica:

spuntosoft@edmaster.it

Web service

Costruire
UN WEB SERVICE IN 5 STEP UTILIZZANDO LA TECNOLOGIA J2EE

La comunità informatica è attualmente concentrata sulle problematiche inerenti la possibilità di poter progettare e realizzare applicazioni che permettano l'interazione tra software realizzate con tecnologie diverse quali .Net e J2EE.

File sul CD
\soft\tools\jwsdp-1_0_01-windows-i586.exe

Cooperazione
Un Web Service si traduce di fatto nella possibilità di far cooperare applicazioni distribuite sulla rete, scritte con tecnologie differenti, basandosi su un protocollo di comunicazione comune a quest'ultime (SOAP).

Esempio ne è stato la Java Conference tenutasi a Milano nel Mese di Maggio 2002, dove il tema principale di discussione del meeting sono stati appunto i WebService e le loro implicazioni nel modo di concepire il Business di una Azienda. Da ciò è scaturito un proliferare di articoli, la cui analisi ha portato alla rilevazione dei vantaggi e degli svantaggi che un tale approccio ha nella costruzione di applicazioni più o meno complesse. Questo articolo, al contrario, si pone come obbiettivo la realizzazione di un concreto esempio di sviluppo di un'applicazione (semplice o complessa che sia) basata sui WebService in ambito J2EE.
Naturalmente essendo la natura dello stesso puramente didattica, è stato improntato alla semplicità, sia dal punto di vista implementativo che di business, per dare modo ai lettori di capire ed assimilare tutti i passi necessari per costruire un'applicazione basata sui Web Service. È comunque importante notare che un Web Service si traduce di fatto nella possibilità di far cooperare applicazioni distribuite sulla rete, scritte con tecnologie differenti, basandosi su un protocollo di comunicazione comune (SOAP). Cosa sta dietro ad un Web Service, ovvero quale tecnologia è stata utilizzata o quanto sia complicata l'applicazione con cui bisogna interagire, non deve essere visibile a chi utilizzerà il servizio pubblicato. Si comprenderà quindi la necessità di un'adeguata strutturazione architetturale del Web Service layer per evitare che la complessità dell'applicazione sottostante abbia come conseguenza un incremento della complessità delle interfacce esposte e quindi dell'implementazione e dell'utilizzo del servizio per l'eccessivo numero di interazioni necessarie. I Web Service, incapsulando la business logic ed i business data ed esponendo di questi soltanto le interfacce di alto livello, devono essere strutturati secondo il pattern *Facade* secondo lo stesso principio utilizzato nel design di un'applicazione J2EE (*http://java.sun.com/blueprints/corej2eepatterns/Patterns/SessionFacade.html*)

IL CASO DI STUDIO

Primo passo nella definizione della nostra applicazione è la descrizione dell'ambito di esecuzione e delle funzionalità che ad essa vogliamo assegnare. Immaginiamo quindi di possedere una catena di distribuzione editoriale. Caratteristica di tale catena è la varietà e moltitudine dei fornitori. Caratteristica dei fornitori è la non omogeneità nella piattaforma tecnologica utilizzata, ovvero ogni fornitore può essersi appoggiato per lo sviluppo del proprio business ad una tecnologia piuttosto che ad un'altra (es: .Net, J2EE). Per esigenze di marketing si è deciso di dotare i propri negozi di distribuzione della facoltà di verificare, attraverso Internet ed in tempo reale, la disponibilità dei titoli e la quantità degli articoli presso i magazzini dei propri fornitori. Inoltre si è anche deciso, sempre per motivi di marketing, di sottomettere on line gli ordini ai fornitori e di poterne verificare lo stato. Naturalmente, da una breve analisi dei requisiti e degli obiettivi, il vincolo predominante in una tale architettura di integrazione è la disomogeneità tecnologica in uso, vincolo che può essere ampiamente superato utilizzando per l'appunto i Web Service. Tralasciando il dettaglio implementativo relativo alla realizzazione del business dei fornitori, il nostro obiettivo è realizzare l'interfaccia verso tali servizi ed eventualmente il business della catena di distribuzione editoriale. Il servizio che si intende realizzare è la conoscenza della disponibilità presso i magazzini di un determinato articolo. Per poter soddisfare tale esigenza è

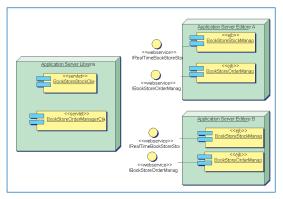



Fig. 1: **Architettura dell'applicazione che permette di conoscere la disponibilità di un articolo nel magazino di un fornitore.**

stata definita una particolare architettura come mostrato nella Fig. 1 e nella Fig. 2. Tale architettura mostra le interazioni tra il client ed il fornitore di servizi, mascherando opportunamente il realizzatore di tali servizi.

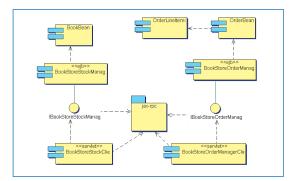



Fig. 2: **L'architettura mostra l'interazione tra il client e il fornitore di servizi, mascherando opportunamente il realizzatore di tali servizi.**

I 5 PASSI CON IL JAVA WEB SERVICES DEVELOPER PACK DI SUN

Poiché il fine ultimo dell'articolo è la realizzazione di un Web Service, si concentrerà l'attenzione su di una organizzazione a punti con alcune parti di codice a corredo. Verrà tralasciata tutta l'informazione relativa alla modalità di gestione di un Web Service da parte di un Server. Naturalmente nello studio di tale codice bisogna prestare attenzione alla versione delle librerie utilizzate. Attualmente è disponibile la ver. 1.0 presso il sito della SUN: *http://java.sun.com/webservices/download.html* (disponibile anche sul CD allegato).

1° - DECISIONI IN MERITO AGLI OGGETTI UTILIZZATI PER L'INTERSCAMBIO DEI DATI

Gli oggetti da utilizzare come mezzo di scambio di informazione all'interno di una architettura distribuita ed integrata devono soddisfare determinate esigenze quali:

- Tipi primitivi
- Tipi JString, Boolean, Byte, Double Float, Integer, Long, BigDecimal, BigInteger, Calendar, Date
- Arrays: *int[], String[], BigDecimal[][].*
- Java Beans purchè:
  1. serializzabili;
  2. esista il costruttore di default pubblico;
  3. non implementino *java.rmi.Remote;*
  4. i campi del bean devono essere *JAX-RPC types.*

2° - COSTRUZIONE DI UNA INTERFACCIA DI TIPO REMOTO PER LA PUBBLICAZIONE DI UN SERVIZIO

È necessario implementare, nella costruzione di un Web Service, una interfaccia remota che permetta la pubblicazione dei servizi.
Naturalmente essendo una interfaccia di tipo remoto è necessaria la gestione delle *Remote Exception*.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import it.ow.softech.transfer.lib.util.LibriTransfer;
public interface IRealTimeBookStoreStock extends
                                          Remote {
  public LibriTransfer[] findCatalogo(String titolo)
                             throws RemoteException;
}
```

3° - COSTRUZIONE DELLA CLASSE CHE IMPLEMENTA L'INTERFACCIA

Lo scopo della costruzione di tale classe è la realizzazione dei servizi che si vuol rendere disponibili. Nel realizzare tali servizi, si può prevedere un utilizzo atomico, ovvero ogni richiesta di servizio è una operazione isolata; oppure un utilizzo composito, il che presuppone che la richiesta di un servizio preveda una interazione articolata in diverse fasi. Per motivi di pulizia architetturale, a seconda delle esigenze, è consigliabile una soluzione mista, anche perché risulta fortemente deleterio progettare un'applicazione basata sui Web Service che preveda

Web service

Costruire
un WebService in 5 passi utilizzando la tecnologia J2EE

Facade (facciata)

Seguire un pattern Facade significa fornire una interfaccia al sistema che ne nasconda la complessità e ne esponga una "facciata" che ne semplifichi l'utilizzo da parte di componenti client.
Così facendo gli elementi esterni non vedono più le singole componenti del sottosistema bensì la Façade, col risultato che gli eventuali cambiamenti interni al sottosistema sono trasparenti all'esterno fintanto che non cambiano i servizi forniti dalla Façade. Fornire un'unica interfaccia di accesso a un sistema complesso evita le dipendenze sulla struttura interna del sistema. Inoltre aiuta a separare il sistema dai clienti esterni(persone o clienti), garantendo una maggiore manutenibilità del progetto, ma non può impedire l'accesso diretto alle classi del sottosistema, anche se ciò è fortemente sconsigliato.

Web service

Costruire
un WebService in 5 passi utilizzando la tecnologia J2EE

un continuo scambio di informazioni via rete quando tale aspetto può essere evitato. Poiché la classe opera lato server, non è necessaria nessuna modalità di richiamo particolare degli oggetti che si vogliono utilizzare.

```
import java.io.Serializable;
import javax.xml.rpc.server.ServiceLifecycle;
import java.rmi.RemoteException;
………………..
import it.ow.softech.lib.exception.ServiceException;
public class RealTimeBookStoreStock implements
 ServiceLifecycle, RealTimeBookStoreStock, Serializable {
……………………….
public  RealTimeBookStoreStock() throws
                                    ServiceException {
……………………………
}
public LibriTransfer[] findCatalogo(String titolo) throws
                                    RemoteException {
……………………………….
}
}
```

4° - CONFIGURAZIONE DEL FILE DI MAKE

Terminata l'implementazione di un Web Service, il passo successivo è la costruzione del file di configurazione. Tale file denominato *"config.xml"* permette la definizione dei parametri del Servizio o eventualmente la locazione dove poter recuperare tali informazioni. Di seguito vengono mostrati due esempi dimostrativi con a corredo una breve descrizione dei tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://java.sun.com/xml/ns
                              /jax-rpc/ri/config">
   <service name="BookBrokerService"
      targetNamespace="http:// BookBroker.com/wsdl"
      typeNamespace="httpBookBroker.com/types"
      packageName="BookBroker">
      <interface name=" IRealTimeBookStoreStock "
         servantName=" RealTimeBookStoreStock "/>
   </service>
</configuration>
```

L'attributo "name" sul tag "Service" rappresenta il nome del servizio che vogliamo creare; tale parametro serve anche per la creazione del file *.wsdl*. Il file generato avrà il seguente nome *BookBrokerService.wsdl* e conterrà al suo interno la definizione di tutti i parametri utili per il richiamo delle funzionalità, da noi definite, da parte dei client. L'attributo "packageName" sul tag "Service" rappresenta il package che si vuole assegnare agli Stub e Ties generati in seguito alla procedura di Making. L'attributo "name" sul tag "interface" rappresenta il nome dell'interfaccia da cui prelevare i metodi di business definiti. Il valore assegnato deve essere comprensivo di package se presente.

L'attributo "servantName" sul tag "interface" rappresenta il nome della classe da cui prelevare le implementazioni relative all'interfaccia di cui sopra.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
   xmlns="http://java.sun.com/jax-rpc-ri/xrpcc-config">
   <wsdl name="GestioneLibriCatalogo"
      location="http://owtp24:7001/ServiceCatalogo
                                   /jaxrpc?WSDL"
      packageName="it.ow.softech.serviceWSDL"/>
</configuration>
```

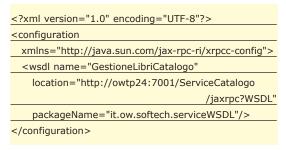
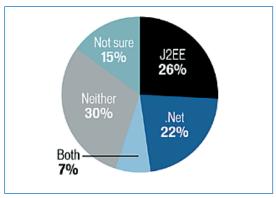



Fig. 3: **I risultati di un'indagine americana sull'orientamento delle aziende in merito alla piattaforma che intendono adottare per lo sviluppo di Web Services. L'indagine è stata compiuta lo scorso anno, è comunque interessante notare il grande equilibrio fra J2EE e .NET.**

L'attributo *"name"* sul tag *"wsdl"* rappresenta il nome del file **.wsdl* da interrogare per il recupero delle informazioni. Su tale file è anche basato l'uso dei registri *"UDDI"*. L'attributo *"location"* sul tag *"wsdl"* rappresenta la locazione fisica dove recuperare il file di cui sopra. L'attributo *"packageName"* sul tag *"wsdl" "* rappresenta il package che si vuole assegnare agli Stub e Ties generati in seguito alla procedura di Making.

5° - ESECUZIONE DELLA PROCEDURA DI MAKING BASATA SUL CONFIG.XML

Per effettuare l'operazione di "making" bisogna utilizzare alcuni tool specifici che permettono la creazione automatica di tutti i file Stub, Ties ed eventuali altri file necessari.

Gli Stub e i Ties permettono la comunicazione tra client e server come se fosse un servizio RMI. SUN fornisce automaticamente un suo

Making

Per effettuare l'operazione di "making" bisogna utilizzare alcuni tool specifici che permettono la creazione automatica di tutti i file Stub, Ties ed eventuali altri file necessari. Gli Stub e i Ties permettono la comunicazione tra client e server come se fosse un servizio RMI.

tool di cui si darà una descrizione del funzionamento di seguito.
Possono essere utilizzati anche altri tool, di cui bisogna comunque analizzare le specifiche, per conoscerne la configurazione, e quindi il modo di utilizzo. Tutti i tool, anche quelli appartenenti a tecnologie diverse, hanno una feature che permette l'utilizzo del *.wsdl per la creazione dei servizi. I file generati sono basati sull'interfaccia, la classe implementante il business e gli oggetti di comunicazione descritti prima, di conseguenza ogni volta che si va a modificare le firme di uno solo di quegli oggetti bisogna effettuare nuovamente l'operazione di making.
Le istruzioni per utilizzare il tool della SUN sono le seguenti:

- **rem** - Ricordarsi di inserire nel classpath tutte le classi e/o librerie necessarie per la creazione del WebService.
- **rem %1 -** Serve per specificare che tipo di *Ties /Stub* deve creare. *--both/server/client--*
- **rem %2 -** La directory ove inserire le classi generate.
- **xrpcc -%1 -classpath .\classes -d %2 config.xml**

I passi da compiere per la costruzione di un Web Service sono conclusi; come si può notare la difficoltà delle operazioni è minima, come minima è anche la complessità di utilizzo di un Servizio da parte del client. Di seguito vengono mostrate le istruzioni necessarie per effettuare l'interrogazione del Servizio implementato negli step di cui sopra. Da notare la presenza di un altro oggetto, tale oggetto viene generato automaticamente e permette il recupero dell'interfaccia Stub.

```
import it.ow.softech.transfer.lib.facade.IRealTimeBook
                                        StoreStock_Stub;
import it.ow.softech.service.StoreStock_Impl;
public class BookStoreStockClient
{
  private IRealTimeBookStoreStock_Stub
                                      stub_Server_1 = null;
public void init() .. {
         ..........................
stub_Server_1 = (IRealTimeBookStoreStock_Stub)(new
    StoreStock_Impl().getIRealTimeBookStoreStock());
stub_Server_1._setProperty(
   javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
                    "http://....../ServiceBookStock/jaxrpc
                          /IRealTimeBookStoreStock");
..........................
}
................
{
  .................
  LibriTransfer[] libri_1 = stub_Server_1.findCatalogo(
                                      parametro_Ricerca);
  ................
}
}
```

CONCLUSIONI

Si può affermare che l'innovazione portata dai WebService sta tutta nella sua semplicità di utilizzo e nella profonda capacità di innovare portando ad un livello sempre più alto l'integrazione fra i sottosistemi.

Ing. Nicola Pepé
Senior Consultant ObjectWay

Web service

Costruire
un WebService in 5 passi utilizzando la tecnologia J2EE

Sul Web

Gli immancabili riferimenti SUN:
java.sun.com/j2ee/
java.sun.com/webservices/

Una panoramica esauriente del meglio della Rete:
http://www.javaskyline.com/webservices/

Il sito ufficiale:
http://www.ws-i.org/

Per tenersi sempre aggiornati:
http://www.webservices.org/

La visione di Borland:
http://www.borland.com/jbuilder/webservices/

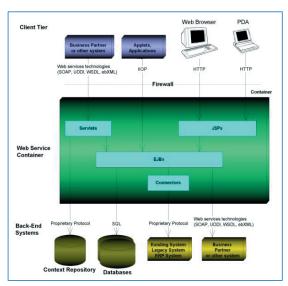
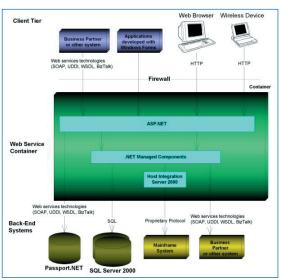



Fig. 4: **Un confronto fra i modelli di implementazione per i Web Services adottati da .NET e J2EE**

La nuova
FRONTIERA DI DIRECTX9

DirectX

Dopo una lunga fase di gestazione, Microsoft ha finalmente dato alla luce l'ultimissima release del suo pacchetto integrato per la gestione del multimedia su piattaforma Windows. L'articolo è diviso in due sezioni, una parte dedicata a chi è alle prime armi nel campo della programmazione 3D, ed un'altra dove saranno trattati argomenti più avanzati. Prima di iniziare però, diamo uno sguardo alle innovazioni più rilevanti.

La nuova release dell'SDK comprende alcuni nuovi tools come ad esempio il *D3DSpy*, un'utility per monitorare il traffico di chiamate verso i componenti DirectX, utilissimo per trovare errori e colli di bottiglia dei nostri programmi. Ci sono dei miglioramenti al Mesh Viewer e al Texture Tool. Ad ogni modo fareste bene a scaricare anche i tools sul sito degli sviluppatori di NVidia: ci sono dei plugin speciali per 3D Studio Max, Maya e Photoshop (per gestire le texture compresse nei formati DXT). Le nuove features sono strettamente legate alle nuove schede DX9 e quindi la ATI Radon 9700 (e simili) e la Nvidia GeForceFX, i prodotti di punta delle rispettive società. C'è anche un'altra scheda di classe DX9, la Matrox Parhelia ma la scarsezza dei suoi driver non la rendono appetibile allo sviluppatore di giochi. Ovviamente DirectX9 può girare anche su schede meno recenti, ma come è ovvio, non possono sfruttarne tutte le potenzialità. Le schede DX9, infatti, implementano il nuovo linguaggio Pixel Shader 2.0, che estende notevolmente i precedenti (1.1, 1.3 delle GeForce ed 1.4 delle Radeon) ampliando sia il numero sia la tipologia d'istruzioni "per fragment". È stato aggiunto il supporto per l'hardware occlusion culling: con DX9 si possono "domandare" all'acceleratore i risultati del rendering. In questo modo potremmo disegnare dei poligoni vicini alla camera, poi renderizzare dei bounding box, degli oggetti in lontananza e chiedere all'acceleratore se è stato scritto qualche pixel. Se ciò non accadesse vuol dire che il box non è visibile, quindi possiamo tranquillamente saltare il rendering di tutto ciò che è contenuto in esso (provate ad immaginare quanti poligoni evitiamo di disegnare inutilmente). È stata ripristinata la funzione di interazione tra GDI e Direct3D (presente in DirectX7 che era stata tolta in DirectX8), in questo modo possiamo usare le funzioni GDI di windows anche nelle applicazioni DirectX a tutto schermo, mostrare dialog e via dicendo. Bisogna stare attenti con questa feature perché influisce negativamente sulle prestazioni. Presenta inoltre alcune limitazioni: per esempio non si può abilitare il *FullScreenSceneAntiAliasing* e funziona solo su alcuni formati di texture, fallendo su quelle con canale alpha. È stato abilitato lo *ScissorTest* ed il supporto per il multihead. Anche D3DX è stata migliorata, offre una nuovissima gamma di oggetti per la gestione delle mesh in skinning con interpolatori personalizzati. Questi oggetti, in congiunzione con gli esportatori in formato .X, presenti per i pacchetti di grafica più diffusi (3DSMax, Maya ma anche l'ottimo Milkshape3D) rendono molto semplice la creazione di personaggi 3D animati.

.NET

Una delle features più attese di DirectX9 è la possibilità di programmare DirectX tramite linguaggi .NET come ad esempio il C# o il VB.NET. Nel caso decidiate di fare questo, vi anticipo che soprattutto usando il C# le performance non sono cattive come possiate pensare. Alla Microsoft hanno dichiarato che le performance si mantengono sul 98% rispetto a quelle del C++, dai miei test non metterei la mano sul fuoco riguardo questa percentuale, ma possiamo usare il C# per applicazioni non "realtime critical". Con questo voglio dire che potreste tranquillamente usare il C# per programmare un editor del vostro motore (avendo tutti i benefici di .NET, i windows.forms, gestione automatica della memoria etc etc) e allo stesso tempo nessuno vi vieta di usare C++ o ManagedC++ per la parte "core" dell'engine. Programmare in C# porterebbe ad altri benefici: vi ricordo che gli oggetti scritti in linguaggi .NET possono essere usati da un qualunque altro linguaggio .NET. Questo vuol dire che se scrivete la vostra logica di gioco in C# ad esempio, avete "gratis" lo scripting per il vostro motore, con esposti tutti i metodi e gli oggetti che avete programmato…Non entro nei dettagli di quest'argomento ma come potete immaginare come sia molto interessante poter fare scripting del proprio engine in C#, VB.NET

File sul CD
\soft\codice\Directx9

Regola della vite

Immaginate di avvitare una vite usando il verso che porta il vostro edge1 all'edge2 secondo l'angolo più piccolo da essi sotteso. Se questo verso è tale da far entrare la vite nella madrevite allora il vettore risultante del prodotto vettoriale è entrante, altrimenti è uscente. Un'ultima cosa: storicamente Direct3D utilizza un sistema di assi cartesiani "left-handed" (letteralmente a mano sinistra) che quindi non segue la regola della mano destra. La libreria di supporto D3DX però ha tutte le doppie versioni delle funzioni geometriche, quelle con suffisso RH usano la regola della mano destra, quelle con suffisso LH non la usano.

La nuova
Frontiera di DirectX9

Sul Web

• La casa di DirectX
http://msdn.microsoft.com/directx

• Un'interessante e polemica riflessione sui linguaggi di shading
http://www6.tomshardware.com/graphic/20021004/index.html

• Due siti MUST su DX9:
http://developer.nvidia.com
http://www.ati.com/developer

• Sito con Tutorials su DirectX8 in italiano (validi anche per DX9)
http://www.gameprog.it/pagghiu

• Mailing List Ufficiale di DirectX. Se avete un problema qui c'è la soluzione
http://discuss.microsoft.com/SCRIPTS/WA-MSD.EXE?A0=DIRECTXDEV&D=0&I=-3

• Librerie DirectX convertite per compilatori Borland
http://www.gameprog.it/pagghiu/roba/borland.htm

• Se volete creare dei video delle vostre demo 3D (magari per farle vedere a chi non ha una scheda di ultima generazione)
http://www.fraps.com/

o quant'altro. Dubito inoltre che qualsiasi linguaggio di scripting "fatto in casa" possa essere più veloce di C#. Ho deciso di usare il C++ per la scrittura degli esempi di questi articoli, ma voglio lanciare un sondaggio tra i lettori di questa rubrica: mandate a *ioprogrammo@edmaster.it* la vostra preferenza sul C# o il C++ per questi articoli ed eventuali argomenti di vostro interesse da trattare (sempre riguardo DirectX9 ovviamente). Volevo aggiungere infine che per chi usa il VisualStudio.NET la microsoft ha messo a disposizione un per poter fare il debugging degli shader. Potete mettere breakpoints, vedere registri, vedere il risultato di espressioni come in un vero debugger. Per installare questo plugin dovete selezionare "Visual Studio Extension" tra i componenti dell'installazione.

PARTE BEGINNER

Prima di cominciare vorrei precisare che mi sembrava poco rispettoso nei confronti dei lettori sprecare questo prezioso spazio per parlare delle inizializzazioni video di un programma DirectX. Provo a motivare questa mia scelta:

1) Sono argomenti molto semplici e noiosi, la documentazione dell'SDK è chiarissima a riguardo e la rete e zeppa di tutorial (e purtroppo solo su queste).

2) Nell'SDK di Microsoft sono dei tutorials ufficiali.

3) Avete a disposizione gli appwizard per VisualC++ 6 e VisualC++.NET che si installano automaticamente con l'SDK. Gli appwizard generano automaticamente il codice "comune" di gestione delle finestre e le inizializzazioni.

4) Ulteriori tutorial sono disponibili sul mio sito personale *http://www.gameprog.it/pagghiu*. Vedere nella sezione "*Tutorials*" quello riguardo "*Inizializzazioni e schermo*". Ci sono anche altri tutorial su argomenti riguardo il 2D che non saranno trattati in questa sede. Il tutto corredato da sorgenti commentati. Si parla di DirectX8 ma convertirli a DirectX9 è questione di fare un "cerca e sostituisci" nei file per cambiare tutti gli "8" con i "9" e cambiare qualche parametro.

In questo articolo cercheremo di capire e utilizzare i vari spazi che Direct3D ci mette a disposizione. Per rappresentare un oggetto in tre dimensioni in Direct3D si utilizzano generalmente dei triangoli nello spazio. Un insieme di triangoli che formano una certa "immagine" nello spazio è generalmente chiamato *Mesh*. Possiamo disegnare un triangolo semplicemente scrivendo le coordinate cartesiane dei tre vertici che lo compongono. Ricordiamo però che non ha senso parlare di coordinate cartesiane se non si fissa un riferimento cartesiano. Prendiamone uno a caso e creiamo un cubo centrato in esso. Potremmo specificare i triangoli che lo compongono uno per uno, ma per fortuna esiste una funzione di D3DX (la libreria ufficiale di estensione a Direct3D di Microsoft) di nome *D3DXCreateBox(…)* che crea un oggetto *ID3DXMesh* contenente i triangoli del nostro cubo, la nostra mesh. Se provassimo a disegnare questa roba così com'è, vedremmo poco o niente. Questo perché noi come "spettatori" ci troviamo al centro del sistema di riferimento e siamo quindi "dentro" il cubo. L'esempio sul CD chiamato "*ComeUsareObjectSpace*" vi fa vedere esattamente questo. Volevo fare due precisazioni, per rendere evidente il fatto che siamo "centrati" nel sistema di riferimento del cubo ho dovuto creare un cubo di lati 1,1,5 (e quindi un parallelepipedo), allungandolo nella direzione della nostra osservazione (l'asse delle Z). Poi ho animato una luce direzionale ed ho invertito la modalità di culling dei triangoli con *SetRenderState (D3DRS_CULLING, D3DCULL_CW)*. È necessario qualche chiarimento su questa cosa. Abbiamo detto che un triangolo è formato da 3 vertici. Se non sono allineati o coincidenti, essi giacciono su un piano comune. Possiamo calcolare la normale a tale piano (e quindi la normale al triangolo), facendo il prodotto vettoriale (cross product) tra due vettori rappresentanti dei lati *(edge)* del triangolo.

Come spero tutti sappiano, il prodotto vettoriale tra 2 vettori restituisce un altro vettore perpendicolare a quelli dati. Dato che i due "edge" giacciono sul piano del triangolo, la normale sarà perpendicolare a tale piano. Ricordiamo però che prodotto vettoriale è anticommutativo, e quindi è importante l'ordine con il quale lo eseguiamo.

In pseudocodice:

```
Vertici v0,v1,v2;     //il nostro triangolo
Vector edge1=v1-v0;
Vector edge2=v2-v0;
Vector normal1 = edge1   x   edge2;
Vector normal2 = edge2   x   edge1;
```

In questo caso risulta che *normal1* ha verso opposto a *normal2*. I flag *D3DCULL_CW* e *D3DCULL_ CCW* (che stanno per *clockwise* e *counter-clockwise*, orario e antiorario) indicano il verso della normale che Direct3D usa per il culling, usando la regola della mano destra (o regola della vite).

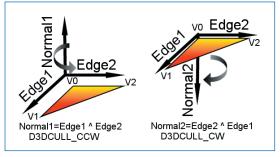



Fig. 1: **Le convenzioni di Orientamento della Normale di DX9"**

La nuova
Frontiera di DirectX9

PROSPETTIVE FUTURE

Se prendete un oggetto e lo posizionate lontano da voi lo vedete più piccolo rispetto a quando ve lo mettete sotto il naso. Quindi, all'aumentare della "distanza" (che per semplicità ora rappresentiamo come coordinata Z dei nostri vertici) diminuiscono le dimensioni dell'oggetto. Potremmo ipotizzare una trasformazione del genere per "proiettare" i vertici sullo schermo:

```
x_a_schermo = x/z;
y_a_schermo = y/z;
```

Sebbene tutto questo funzioni, in pratica si utilizzano delle trasformazioni molto più complicate, che per questioni di spazio non sto qui a raccontarvi. Il motivo di questa complicazione è il clipping. Se un triangolo proiettato risulta parzialmente fuori schermo,bisogna "tagliarlo" in qualche modo per poterlo disegnare. Le matrici di clipping usano degli spazi particolari che rendono molto semplice questa operazione. Quello che vi interessa sapere è che la matrice prospettica rappresenta il vostro "cono di proiezione", tutto quello che è al di fuori di esso non verrà riportato a schermo. Per essere precisi dovete pensare a questo cono come ad una piramide tronca a base quadrangolare, la cui altezza coincide con la vostra direzione di osservazione. La base minore di questa piramide (detta *near clip plane*) rappresenta sostanzialmente il vostro schermo, mentre la base maggiore (detta *far clip plane*) è il piano limite oltre il quale non viene disegnato più nulla. Ci sono vari modi per specificare questo tronco piramidale. Potremmo specificare l'altezza e l'angolo del vertice oppure la distanza tra i due piani (*far* e *near*) e i lati della base di uno di essi, o altro ancora. Nel nostro esempio facciamo così:

```
D3DXMATRIX matProj;
float aspect = (float)g_width/(float)g_height;
D3DXMatrixPerspectiveFovLH(&matProj, D3DX_PI/3.0f,
                                        aspect ,0.1f, 50.0f);
g_device->SetTransform(D3DTS_PROJECTION,&matProj);
```

Impostiamo un angolo al vertice di pigreco/3 (60 gradi), con un rapporto tra i lati della base piramidale dato dal parametro *aspect*, e *0.1f* la distanza del *near plane* dal vertice e *50.0f* la distanza dal *far plane* dal vertice. Non specificate mai *0* come distanza del *near plane*, altrimenti non vedrete nulla a schermo (ci sarebbero delle divisioni per zero nella fase di proiezione).

WORLD SPACE

Se volessimo animare il cubo nel tempo, dovremmo trovare un modo di disegnarlo in posizioni diverse durante istanti di tempo successivi. Potremmo specificare i vertici del nostro cubo a mano, spostandoli dove ci interessa ma tutto ciò è complicato ed inefficiente. Per fare questo applichiamo una trasformazione al nostro cubo, indicandone la posizione e l'orientazione

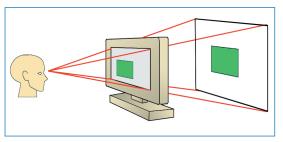



Fig. 2: **Il tronco ("Frustrum") di proiezione.**

"nel mondo" (tridimensionale). Nell'esempio di prima l'object space (spazio di definizione del parallelepipedo) e il world space (spazio dove esso è posizionato ed orientato) coincidevano. Per questo vedevamo il suo interno. Una trasformazione in Direct3D può essere rappresentata da una matrice. Indichiamo a Direct3D di posizionarla usando il metodo *g_device->SetTransform(…)* col parametro *D3DTS_WORLD* e l'indirizzo della matrice contenente la trasformazione. Vi ricordo che la matrice che passate a Direct3D può essere la risultante di tante trasformazioni successive (ad esempio una traslazione, seguita da una rotazione,seguita da una scala etc etc). Queste trasformazioni possono essere concatenate usando il prodotto "righe per colonne" tra matrici. In Direct3D questo si fa usando *D3DXMatrixMultiply* oppure l'operatore "*" (è utilizzata la tecnica dell'overloading per le matrici D3DXMATRIX). Attenti, il prodotto tra matrici è anticommutativo quindi l'ordine di moltiplicazione è importante!

L'esempio *"ComeUsareWorldSpace"* disegna due volte il cubo con colori diversi (tra l'altro usando sempre la stessa mesh) in uno stesso frame, ed anima le posizioni nel tempo (utilizzando funzioni di seno e coseno con il tempo come parametro).

VIEW SPACE

Immaginate un'altra situazione: avete un insieme di oggetti posizionati nello spazio (cambiando di volta in volta la matrice WORLD). Volete andarevene in giro attraverso questi oggetti, come succede in un qualunque gioco tridimensionale, come fate? Con quello che sapete fino ad ora dovreste trasformare tutti gli oggetti della scena con delle matrici world invertite. Esiste un'alternativa, il *View Space*. Introduciamo questo nuovo spazio, che rappresenta una sorta di "telecamera" nello spazio, la nostra posizione in quanto osservatori. Impostiamo una matrice per il *ViewSpace* utilizzando il metodo *g_device->SetTransform (…)* col parametro *D3DTS_VIEW* e l'indirizzo della matrice contenente la trasformazione. Le funzioni per creare *View Matrix* sono quelle del tipo *D3DXMatrixLookAtLH*.

L'esempio *"ComeUsareViewSpace"* differisce dal precedente in quanto la view matrix e il nostro tronco di visualizzazione, associato con la projection matrix.

Ho aggiunto il disegno di una griglia fissa, per far capire meglio che oltre a muoversi i due cubi, ci muo-

Sul Web

- **Un OTTIMO sito in italiano con dei tutorials eccellenti, soprattutto sugli spazi usati in computer graphics**
  http://www.nablasoft.com/
- **Papers della nvidia sul projective texture mapping**
  http://developer.nvidia.com/view.asp?IO=Projective_Texture_Mapping
  http://developer.nvidia.com/view.asp?IO=projective_textures
  http://developer.nvidia.com/view.asp?IO=texgen_tex-matrix
- **Come applicare la projective texture mapping alle ombre**
  http://developer.nvidia.com/view.asp?IO=gdc_projshadows

DirectX

La nuova
Frontiera di DirectX9

viamo anche noi osservatori nello spazio.
Vediamo la griglia spostarsi, infatti, pur essendo quest'ultima fissa nello spazio.

Fig. 3: **Gli spazi di Direct3D. Visti "da fuori" in (a) e (b) e visti dalla telecamera in (c) e (d).**

PARTE ADVANCED

Esaminiamo un'applicazione più avanzata: il *Projective Texture Mapping* (da ora in poi *PTM*). Con questa tecnica è possibile proiettare una texture su scene poligonali arbitrarie. Potete immaginare di avere attaccato ad un proiettore, come quelli usati nei circhi o negli studi televisivi, un foglio di carta lucida con qualcosa disegnato sopra.
L'esempio *"ProjectedTexture"* implementa questa tecnica. Questo esempio utilizza gli *"Effect File"* di cui potete trovare informazioni sulla documentazione dell'SDK.
L'idea alla base del PTM è banale, bisogna solo stare attenti con gli spazi in cui lavorare. Utilizziamo per ora solo la *Fixed Function Pipeline* (da ora in poi FFP), senza vertex shader o HLSL.
Tutto quello che Direct3D in FFP può fare per noi è:

1) Copiare la posizione di un vertice nel suo set di coordinate texture.

2) Applicare una matrice di trasformazione alle coordinate texture.

3) Proiettare le coordinate texture.

Esattamente quello che fa il codice di seguito, contenuto nell'effect file:

```
//Dal file "ProjectedTexture.FX"
//...omissis
technique ProjectedTextureOnSphere
{
  pass P0
  {
    //...omissis
    TextureTransform[0] = (tex_matrix);
    TexCoordIndex[0] = CAMERASPACEPOSITION;
    TextureTransformFlags[0] = COUNT3|PROJECTED;
    //...omissis
  }
}
//...omissis
```

Effect File

In breve gli *Effect File* sono degli script in formato testo, compilati da D3DX all'avvio del programma. Questi file ci consentono una più semplice designazione degli stati necessari al rendering e possono anche contenere script HLSL come vedremo avanti.

CAMERASPACEPOSITION fa quanto descritto nel punto 1. I vertici sono però in view space, come si evince dal nome di questo flag. Dobbiamo quindi trovare una matrice che migri da "view space" allo spazio del nostro "proiettore virtuale" e da qui alle coordinate texture vere e proprie, proiettando queste coordinate texture 3D. Riprendendo i paragrafi precedenti vi renderete conto di come non abbiamo fatto altro che prendere i vertici dei nostri triangoli nello spazio (in 3D), trasformarli a nostro piacimento e proiettarli sullo schermo (che è 2D). Il risultato della proiezione era un punto sullo schermo. Quello che facciamo ora è pressoché simile solo che invece di ottenere in output un vertice proiettato sullo schermo, otteniamo un vertice proiettato sulla texture, cioè una "coordinata texture". Il tronco del nostro "proiettore virtuale" è dello stesso tipo di quello della camera attraverso la quale "vediamo" il nostro mondo 3D. La prima cosa da fare è portare il nostro vertice dallo spazio della camera allo spazio "world". Vi ricordo, di nuovo, che questa coordinata texture "3D" è stata generata con *CAMERASPACEPOSITION*. Per fare ciò usiamo l'inversa della matrice view (o camera).
A questo punto abbiamo il vertice in World Space: ora dobbiamo trasformarlo nello spazio "proiettore". Questo spazio è rappresentato da una matrice con la sua posizione e l'orientamento. Non ci resta che usare una matrice di proiezione prospettica, per trasformare i vertici nel projection space. La proiezione vera e propria la esegue Direct3D perché abbiamo usato come *TextureTransformFlags= COUNT3|PROJECTED*. Questi flag stabiliscono che il device aspetta in input un set di coordinate texture tridimensionali (*COUNT3*), che proietterà (*PROJECTED*) dopo averle applicato una matrice di trasformazione (*TextureTransform*). La proiezione consiste nel dividere le due componenti per la terza cioè (x/z, y/z), producendo la tanto sospirata coordinata texture del vertice.
Se si riflette un attimo, è esattamente quanto descritto nel paragrafo *"Prospettive future"*: al posto dello schermo abbiamo una texture. Prima di cantar vittoria dobbiamo correggere una piccola discrepanza tra il sistema di riferimento dello *"spazio texture"* e quello dello "spazio del proiettore" che abbiamo usato ora. Le matrici create con *D3DXMatrixPerspective* (e quindi anche quella del proiettore) trasformano il contenuto del tronco di proiezione in un valore compreso nel quadrato di vertici *(-1,-1), (-1,1), (1,-1), (1,1)* estruso a partire da un piano con $Z = -1$ fino a $Z = 1$. In genere questo spazio viene chiamato *"Cuboid Space"* (è effettivamente un cubo). I vertici della texture invece hanno coordiante *(0,0) (0,1) (1,0) (1,1)*. Dobbiamo quindi trovare una matrice che migri dal *"projection space"* al *"Texture space"*, prima che avvenga la proiezione vera e propria con il flag *PROJECTED*. Creiamo allora la matrice *lightScale*, che scala tale quadrato di 0.5 nelle

due direzioni (quindi nell'intervallo *[-0.5, 0.5]*) e lo trasla in alto a sinistra ancora di 0.5 (quindi *[0, 1]*). Trasliamo anche sulla *Z* di 1 come potete leggere dai listati, in modo da farlo giacere su *Z=0*, che è il *"near plane"* del proiettore. La *Z* ovviamente è da intendersi relativa allo spazio del proiettore. Essa è la distanza di un punto misurata a partire dal near plane del proiettore.Concateniamo tutte queste matrici con il prodotto righe per colonne *(* in D3DX)* in una matrice finale *"matTex"e* la passiamo a *SetTransform (D3DTS_TEXTURE0).*

```
//Dal file "ProjectedTexture.cpp"
float aspect = (float)g_width/(float)g_height;
D3DXMATRIX matProj,matView,matWorld,matTex;
D3DXMATRIX invWorld,invView,lightScale;
D3DXMatrixPerspectiveFovLH(&matProj,...); //parametri
                                              a piacere
D3DXMatrixRotationZ(&matWorld,sinf(t));
D3DXMatrixLookAtLH(&matView,...); //parametri a piacere
D3DXMatrixInverse(&invView,0,&matView);
D3DXMatrixInverse(&invWorld,0,&matWorld);
D3DXMatrixPerspectiveFovLH(&g_lightProj,...);
                                   //parametri a piacere
D3DXMatrixTranslation(&lightScale,0.5f,0.5f,1);
lightScale._11 = 0.5f;
lightScale._22 = -0.5f;
lightScale._33 = 0;
D3DXMatrixLookAtLH(&g_lightView,...);
                                   //parametri a piacere
matTex = invView*g_lightView*g_lightProj*lightScale;
```

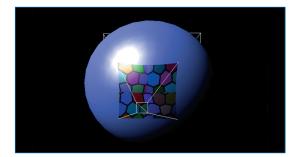



Fig. 4: **L'esempio "ProjectedTexture" in azione.**

Se provate a far partire l'esempio *"ProjectedTexture"* vedrete la texture proiettata, il tronco del proiettore e tenendo premuto spazio vedrete anche in sovrimpressione i poligoni della sfera. Come potete notare il proiettore lavora indipendentemente da come si muove la sfera. Avrei potuto aggiungere una texture *"attaccata"* alla sfera con il multitexturing per rendere più evidente questo fatto. Ho preferito non farlo per motivi di chiarezza del programma. Possiamo sintetizzare i passaggi che portano dalle coordinate di un vertice (3D) alle coordinate della texture su di esso proiettata:

View Space –> World Space –> Camera Space (del proiettore) –> *Projection Space* (del proiettore) –> *Matrice scala* (per "aggiustare" la discrepanza tra i due sistemi di riferimento, quello texture e projection del proiettore).

HLSL

Nel Codice *\soft\codice\directx9\preojecttexture.fx* è mostrata una parte di uno script HLSL che implementa quanto fatto dalla FFP nei passaggi descritti precedentemente.

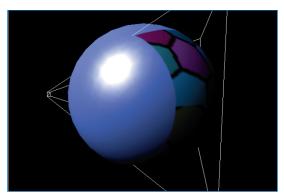



Fig. 5: **La Back-Projection in azione.**

Come potete vedere un programma HLSL è una funzione con dei parametri in entrata, dei quali specifichiamo la dimensione *(float3)* e la semantica *(POSITION, NORMAL* o quant'altro). La semantica serve per "indicare" al compilatore cosa deve mettere in quelle "variabili". Questo script è una sorta di "callback" di Direct3D, chiamata una volta per ogni vertice. Lo script HLSL deve produrre per ogni vertice passato, un valore in output, senza accedere alle informazioni degli altri vertici. Quello che facciamo è trasformare il vertice in view space (quindi mettendoci nelle stesse condizioni di CAMERASPACEPOSITION di prima), trasformarlo con la *tex_matrix* e proiettarlo, dividendo le prime due componenti per la terza. Il risultato finisce direttamente nelle coordinate texture del vertice. Nella tecnica *"ProjectedTextureOnSphereHLSL"* dell'Effect File ci basta scrivere.

```
VertexShader = compile vs_1_1 VS();
```

Per trasformare lo script in qualcosa di sensato per il nostro acceleratore. Le parti omesse dello script HLSL sono quelle relative al calcolo dell'illuminazione diffusa e speculare della nostra sfera. Questo proiettore è un po' "anomalo" perché proietta anche sui poligoni che non sono orientati nella sua direzione. Questo fenomeno viene chiamato *"BackProjection"*, esistono vari metodi per eliminarlo. Mi spiace di non poter commentare ulteriormente quest'ultima sezione sull'HLSL ma lo spazio a nostra disposizione è esaurito. Sicuramente questi argomenti saranno ritrattati più nel dettaglio in futuro, ma l'intenzione era di dare solo un assaggio della potenza dell'HLSL in quest'articolo d'introduzione alle DirectX. Le novità di DirectX9 sono veramente tante, c'è l'imbarazzo della scelta su quali scegliere e su quali sperimentare.

Stefano Cristiano

DirectX

La nuova
Frontiera di DirectX9

HLSL

La misteriosa sigla HLSL sta per "High Level Shading Language", esso è l'evoluzione dei vertex e pixel shader. Siete stanchi di ricordare a memoria i registri nei quali avete passato le vostre costanti di shading? Siete stanchi di leggere il criptico codice simil ASM dei vostri pixel o vertex shader? Con L'HLSL potete scrivere i vostri shaders in un linguaggio simile al C, e compilarlo per qualunque versione di vertex o pixel shader v'interessi. Se avete sentito parlare del "CG" cioè il "C for computer Graphics" della NVidia sappiate che sono quasi la stessa cosa, o meglio hanno lo stesso obiettivo,semplificare lo sviluppo degli shaders nelle nuove pipeline programmabili. Ora come ora è possibile compilare i programmi HLSL con il compilatore CG (non il contrario), ma non è detto che lo sarà in futuro, perché non esistono accordi "ufficiali". È in corso una guerra sui newsgroup e mailing list di programmazione 3D sui due linguaggi. Insomma la prospettiva dello sviluppo shaders è in continua evoluzione, vedremo cosa succederà.

Macromedia
COMMUNICATION SERVER

Flash

Il vostro capo vi ha chiesto di progettare e realizzare un sistema di videoconferenza per la vostra azienda, e voi, pur non sapendo da dove cominciare, avete risposto con aria sicura di non avere nessun problema a riguardo e di poter soddisfare le sue esigenze in breve tempo.

In casa Macromedia ultimamente sono venuti alla luce due nuovi prodotti che rivoluzionano il modo di programmare applicazioni di rete, consentendo al nuovissimo FlashMX di poter disporre, oltre che dei suoi noti e potenti strumenti, anche di importanti possibilità quali lo streaming testo/audio/video in real-time, nonchè della possibilità di essere interfacciato con applicazioni costruite per lavorare con i principali ambienti server oggi disponibili: java2EE, .NET ed ovviamente ColdFusionMX. I nomi di queste due nuove tecnologie sono rispettivamente:

- **Communication Server MX**, che si occupa di fornire a FlashMX un mezzo per configurare ed utilizzare dispositivi multimediali per qualsiasi tipo di applicazione immaginabile, anche tempo reale.
- **Remoting MX**, che si occupa di fornire un set di API che interfacciano il nostro client in Flash direttamente con il server Java, .NET o ColdFusion che abbiamo realizzato, senza passare per linguaggi di interscambio tipo XML.

I due prodotti sono perfettamente combinabili e permettono di rendere estremamente intuitiva e veloce la realizzazione di strutture distribuite che abbinino, alla funzionalità necessaria, anche l'attrattiva di contenuti dinamici e multimediali. Sebbene il nostro interesse verta maggiormente sul Communication Server, vediamo di capire come è possibile creare software che sfrutti appieno le possibilità di entrambi gli strumenti, in modo da comprendere a pieno come funziona l'infrastruttura immaginata dalla Macromedia.

Fig. 1: Architettura implementata da Remoting MX.

ARCHITETTURA DISTRIBUITA

Remoting MX rende FlashMX in grado di comunicare direttamente con il server, invocandone metodi e condividendone proprietà alla stregua di tecnologie come RMI o COM. Remoting infatti si preoccupa di rendere disponibili all'interno delle API di Actionscript, il linguaggio interno di Flash, strumenti ed oggetti utili per la gestione di progetti multi-tier, come ad esempio la gestione delle connessione a database, ed un debugger capace di ispezionare anche i comportamenti lato server della nostra applicazione. È esattamente qui che entra in ballo Communication Server MX, che si occupa di un aspetto che, malgrado lo scambio di dati nativo con il server, sarebbe continuato a rimanere ostico per la maggior parte degli sviluppatori: la gestione di contenuti multimediali all'interno di applicazioni su più livelli. La soluzione che la Macromedia ci offre, consiste in un ennesimo strato software che si occupa di gestire i contenuti multimediali in streaming rendendo ad un ipotetico server i dati finali, pronti per essere utilizzati. La possibilità di gestire il flusso di dati in real-time (grazie al protocollo RTMP), nonché la capacità di accettare molte connessioni simultanee, rendono il prodotto in grado di soddisfare tutte le possibili esigenze che l'utenza media può incontrare. Lo strumento, che è un server a tutti gli effetti, può essere utilizzato con o senza il supporto di Remoting MX, ma è insieme che i due raggiungono i risultati migliori. Communication è stato infatti pensato esclusivamente per la presentazione di contenuti in streaming ed è inadatto ad un qualsiasi altro diverso uti-

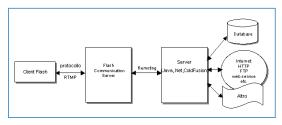



Fig. 2: Una visione d'insieme.

File sul 1º CD
soft\codice
\VideoConferenza.zip

File sul 2º CD
\soft\server
\CommunicationComponent.zip
\soft\server
\FlashCommunicationServer.exe

Multiutenza

Per testare l'esempio riportato nell'articolo aprite pure più istanze del client sullo stesso computer, ma ricordate che questa situazione spesso non viene ben gestita dal prodotto.

Altri server

Sebbene questo esempio non tenga in considerazione la comunicazione dell'applicazione generata con server diversi da Communication, è possibile tramite Remoting interfacciare il nostro programma con servers progettati appositamente. Per maggiori informazioni consultate la documentazione su RemotingMX presente sul sito della Macromedia.

lizzo, mentre Remoting può soddisfare qualsiasi desiderio ma pecca sulla gestione di informazioni multimediali real-time. La struttura dell'applicazione pensata a casa Macromedia si arricchisce quindi di un nuovo livello, che si frappone tra il server ed il client così come in Fig. 2.

COSA OCCORRE

Ecco cosa dobbiamo avere sul nostro computer per poter arrivare a realizzare il programma d'esempio:

- Macromedia Flash MX.
- Macromedia Flash Communication Server MX.

Nota: In fase di installazione di Communication Server MX, vi verrà chiesto se installare il prodotto come *Development* o *Deployment* . L'esempio di questo articolo implica che abbiate scelto l'opzione *Development*.

- Communication Components.
- Una Webcam.

Ora che abbiamo tutto il necessario, è ora di riempire decine di Megabyte del nostro HardDisk dei colori e suoni che la multimedialità ci offre, seguendo le procedure di installazione dei prodotti ottenuti. La giusta sequenza consiste nel lanciare per primo il file di FlashMX, in quanto questo è il fulcro di quanto stiamo per realizzare e, senza esso, nessuno degli altri due file potrebbe funzionare. Scoprirete, una volta fatto doppio click sul file in questione, che non vi è nessun passaggio complicato nell'operazione di installazione del prodotto, che si posiziona nella directory specificata senza porvi alcuna domanda "imbarazzante". Il discorso non vale invece per Communicator, il secondo programma utile per il nostro esempio. Vi verrà infatti richiesto più volte di prendere delle decisioni che potrebbero risultarvi di difficile comprensione e che, proprio per questo motivo, provvederò a chiarirvi subito. Il primo passaggio che richiederà il vostro intervento riguarda l'inserimento di una *user-id* ed una *password* che vi verranno richieste tutte le volte che accederete al prodotto per configurarne le funzioni. Dopo aver provveduto all'immissione dei dati, vi sarà dato di scegliere se intendete utilizzare il prodotto come utente o sviluppatore. Non offendetevi per essere stati trattati come semplici operatori ed optate decisamente la seconda possibilità. La decisione successiva riguar da la possibilità o meno di affiancare il prodotto ad un altro server web risiedente sul vostro computer. Visto che per il nostro esempio non è necessaria questa opzione, scegliete di installare il prodotto, nella versione *stand-alone*, nella cartella del vostro hard-disk che più vi aggrada. Dopo aver preso quest'ultima decisione, il prodotto termina l'installazione ed il vostro computer è quasi pronto per iniziare a lavorare.

Dobbiamo infatti ancora prenderci la briga di fornire alla nostra interfaccia in Flash dei nuovi componenti che possano sfruttare le potenzialità offerteci da Communicator: possiamo farlo installando il set apposito preparato dalla Macromedia. Visto che però questa volta non vi troverete di fronte ad un unico file eseguibile, dovrete compiere delle operazioni per permettere a Flash l'utilizzo dei suddetti componenti. Per procedere dovete copiare il file *"Communication Components.fla"*, presente tra quelli scaricati dal sito Macromedia, nella cartella *\First Run\Components* situata nella cartella di installazione di FlashMX. Questo passaggio permetterà a Flash di disporre dei componenti multimediali che utilizzeremo per sviluppare l'applicazione, ma occorre anche registrare i medesimi sul server perché questi possano essere ben riconosciuti da quest'ultimo. Si deve quindi copiare la cartella *\scriptlib* presente nel pacchetto scaricato, nella directory *\flashcom* risiedente nella directory di installazione di Communicator, sovrascrivendo quella presente. Fatto ciò abbiamo veramente terminato e possiamo dedicarci alla creazione del nostro sistema di video conferenza senza indugiare oltre.

SI COMINCIA

Visto che abbiamo deciso di fare una bella figura con il nostro capo, l'applicazione che svilupperemo sarà completa di tutte le funzionalità che un sistema di videoconferenza può desiderare. Avremo un elemento che ci permetterà di effettuare il *login* e di essere riconosciuto dagli altri utenti con una user-id, avremo ovviamente delle finestre che rappresenteranno ciascuno degli utenti connessi alla videoconferenza e potremo avvalerci anche di un elemento che raffiguri la lista di tutti i partecipanti. Inoltre doteremo la nostra applicazione di un componente che ci permetterà di settare la larghezza di banda desiderata per la ricezione delle immagini, così da poter configurare meglio l'applicazione stessa in base alle capacità della rete alla quale è destinata. Infine posizioneremo un led sulla nostra interfaccia utente, come indicatore di segnale che ci permetterà si identificare immediatamente la bontà della trasmissione. Iniziamo con il chiarire che esistono due possibili strade da percorrere. La prima comporta la scrittura di codice che crei dinamicamente le caratteristiche sopra indicate e che provveda, sempre tramite programmazione, a relazionarle tra loro per fare in modo che cooperino. La seconda consiste nel creare manualmente delle istanze dei componenti che verranno utilizzati e collegarle tramite i nomi che avremo provveduto a fornire. Noi sceglieremo il secondo approccio poiché credo che risulti molto più immediato a chi ha poca dimestichezza con il prodotto, consentendo la realizzazione dell'esempio anche a chi ha installato FlashMX appositamente per questo articolo. Cominciamo quindi con il lanciare FlashMX

Flash

Macromedia Communication Server

Versioni prova

Esistono versioni di prova di ciascuno dei prodotti sopra indicati, scaricabili direttamente dal sito Macromedia. L'indirizzo per ottenere gli eseguibili di prova di FlashMX e Communication Server MX è: *www.macromedia.com/it/software/trial_download/*, mentre è possibile ottenere i componenti Flash necessari a portare a termine il nostro programma d'esempio all'indirizzo: *www.macromedia.com/software/flashcom/download /components/*. Prima di procedere con l'installazione dei componenti, è conveniente fare una visitina all'indirizzo *www.macromedia.com/software/flashcom/productinfo/systemreqs/* per esaminare i requisiti di sistema richiesti.

Aggiornamenti

Macromedia Communication richiede per funzionare correttamente l'ultima versione di FlashMX e del suo plug in. Controllate accuratamente i files README del prodotto ed il sito della Macromedia per eventuali problemi di funzionamento.

Flash

caricando l'interfaccia del prodotto. Per iniziare a lavorare, la prima cosa da fare è visualizzare i componenti che ci occorrono per eseguire il lavoro. Per far ciò aprite il pannello Components dal menù *Window\Components* o premete il tasto *F11*. Una volta fatto, dalla lista a discesa presente sul bordo superiore del pannello scegliete i componenti di Communication, facilmente identificabili dalla scritta "Communication components", e vi appariranno gli strumenti che utilizzeremo a breve. Identificate quello con nome *Simple Connect*, questo componente è alla base del collegamento con il server. Prima di inserire questo oggetto sullo stage, è sicuramente meglio impostare il server per la ricezione di dati da una nuova sorgente. Lasciando quindi aperto Flash in sottofondo, andiamo a registrare la nostra applicazione presso il server.

LATO SERVER

Tutto ciò che occorre per fare in modo che Communication "veda" l'applicazione che stiamo per realizzare è creare una cartella in una specifica directory del server medesimo. La directory in questione è *\flashcom\applications*, che potete trovare dentro il percorso di installazione che avete scelto per Communicator. Create quindi la vostra sottocartella e datele il nome che preferite, ma ricordate che esso dovrà essere il nome che utilizzerete anche per la vostra applicazione Flash. Una cosa importante è infatti nominare alla stessa maniera sia la cartella che il file che prepareremo in FlashMX (file che salveremo al suo interno). L'utilizzo che faremo della cartella, infatti, consisterà nel contenere il file che avevamo iniziato a preparare in Flash e che abbiamo interrotto, oltre ad un file *"main.asc"* di cui ci occupiamo invece immediatamente. Il file main.asc ha il compito di indicare a Communication i componenti che verranno utilizzati nella nostra applicazione. Nel nostro esempio e nella maggior parte dei casi è sufficiente crearlo come un semplice file di testo contenente la stringa *"load ("component.asc");"* e salvarlo nella cartella appena creata. Visto che ora abbiamo la cartella nella quale registrare il file che avevamo iniziato a creare in Flash, torniamo pure su quest'ultimo e salviamolo nella cartella omonima. Nell'esempio, il nome scelto per il file Flash e per la cartella che lo contiene è *"VideoConferenza"*. Una volta salvato il file Flash dentro la cartella, il server è pronto a gestire eventuali connessioni effettuate tramite quella applicazione, e noi passiamo a verificarlo subito trascinando il componente *Simple Connect* sullo stage di FlashMX.

Ora abbiamo a disposizione lo strumento che ci permette la comunicazione con il server, ma dobbiamo segnalare al componente in quale directory è presente l'applicazione alla quale fare riferimento per il flusso di dati in entrata ed in uscita. Selezionate a tal scopo il componente stesso cliccandovi sopra ed aprite il pannello *Properties* dal menù *Window\Components* o premendo *Ctrl-F3*. Aprite la scheda *Parameters*, ed alla voce *Application Directory* immettere il nome che avete scelto per la cartella creata in Communication nonché per il file stesso. Nel farlo, lasciate intatta la prima parte della stringa, cioè *"rtmp:/"*, in modo che il risultato diventi, nel mio caso *"rtmp:/VideoConferenza"*. Dopo aver effettuato questo passaggio il server potrà effettivamente essere raggiunto dal flusso di dati del componente. Facciamo subito una prova verificando che vi sia un effettivo scambio di dati tra client e server aprendo l'interfaccia di Communication.

Fig. 3: **La plancia di comando.**

Nella directory dove avete installato quest'ultimo prodotto potete trovare diversi files, tra cui quelli per avviare ed interrompere il server. Andate quindi nella cartella ed entrate nella sottodirectory *Tools* dove potete trovare il file *"StartServerService.bat"*. Lanciandolo attiverete il server Communication lasciandolo in attesa di connessioni da parte di clients ad una delle applicazioni registrate. Ovviamente l'altro file, *"StopServerService.bat"*, serve ad interrompere il servizio. Bene, dopo aver avviato il server vediamo l'interfaccia del prodotto lanciando il file *"admin .html"* presente nella sottocartella *\flashcom\admin* che trovate come sempre nella directory di installazione del prodotto. Dopo aver effettuato il login immettendo la *user-id* e la *password* che vi sono state richieste durante l' installazione, ciò che appare ai vostri occhi è il pannello di amministrazione di Communication, dal quale potete accedere a tutte funzionalità del programma. Come potrete osservare, nella zona centrale della schermata, che dovrebbe contenere i dati relativi alle connessioni in corso per tutte le varie applicazioni registrate, non vi è alcuna traccia della nostra *"VideoConferenza"*. Questo accade perché effettivamente non vi è alcuna connessione attualmente in corso e quindi Communication non rileva alcun passaggio di informazioni. Per correggere questa situazione e verificare il corretto scambio di dati tra server e client, dobbiamo lanciare almeno un'istanza del filmato che abbiamo realizzato in Flash. Tornate quindi a Flash e pubblicate il filmato che avete realizzato scegliendo *File/Publish* o premendo *Shift+F12*. Potete ora scegliere, per avviare il programma, se lanciare il file *.html* che trovate nella cartella dove avete salvato il filmato Flash che state

VideoConference

È il componente che si occupa di realizzare la videoconferenza vera e propria, gestendo la multiutenza con un sistema a finestre multiple. Ogni utente collegato alla videoconferenza comporterà infatti l'apertura di una nuova finestra nella quale apparirà ciò che la sua webcam sta riprendendo in quel momento.

ConnectionLight

È esclusivamente un led che indica se la connessione è presente o meno e per il quale è possibile settare la latenza nella ricezione dei dati.

PeopleList

È il componente che ci permette di avere sott'occhio tutti gli utenti collegati alla videoconferenza in un determinato momento. Quando un utente si collega all'applicazione, il suo nome viene inserito nella PeopleList.

creando o premere *Ctrl+Invio* e controllare il risultato direttamente da dentro Flash. Quale che sia la scelta che effettuate, vi troverete di fronte ad una schermata bianca con al centro il componente che avete piazzato sullo stage. Se ora effettuate il login inserendo uno user-id di vostra scelta e premendo il tasto *Login*, Communication avvierà effettivamente la nostra applicazione ed inizierà lo scambio di dati con altri eventuali client. Per constatare quanto detto, riaprite il pannello di amministrazione del prodotto e premete sul tasto *Update* presente al centro della schermata. Immediatamente potrete veder apparire il nome della nostra applicazione ed il nostro client collegato… l'operazione ha avuto buon esito. Ora che abbiamo constatato che FlashMX e Communication riescono a "vedersi" è ora di passare alla fase più divertente della nostra applicazione, cioè inserire i componenti che ci permetteranno di realizzare effettivamente la videoconferenza.

LATO CLIENT

Riaprite il file che state realizzando con FlashMX e visualizzate il pannello Components come fatto in precedenza. Dalla lista a discesa, se non gia selezionati, scegliete i componenti di Communication in modo da poterli utilizzare. Bene, iniziamo ad aggiungere tutte le funzionalità di cui il nostro prodotto necessita trascinando sullo stage i seguenti componenti:

- VideoConference
- ConnectionLight
- SetBandWidth
- PeopleList

Disponete i componenti come meglio credete e sarete quasi pronti a far funzionare la vostra applicazione. L'ultimo passaggio consiste infatti nel registrare i componenti che avete trascinato sullo stage presso l'elemento Login che abbiamo inserito per primo. Login è infatti il componente principale, colui che mantiene la connessione e che si occupa di fornire i dati agli altri elementi, per questo deve ricevere in input il nome delle istanze dei componenti che avete inserito nell'applicazione. Il primo passo consiste quindi nel dare un nome a tutte le istanze dei componenti che avete sullo stage semplicemente cliccandovi sopra e, nel pannello *Properties (Ctrl+f3 oppure Window/Properties)*, immettere un nome dove appare il testo <*Instance Name*>.
Il nome deve essere univoco e servirà ad identificare i componenti presso il componente principale *Login*. Fatto ciò selezionate l'elemento *Login* e nel pannello *Properties* selezionate la scheda *Parameters*. Nel componente in questione è possibile settare oltre che l'applicazione server alla quale fare riferimento, anche un'array di nomi di componenti che debbono utilizzare la connessione medesima. Quello che dovete fare per aggiungere i nomi che avete appena assegnato, è cliccare in corrispondenza dell'elemento chiamato *Communication Components* e cliccare ancora sul pulsante con la lente di ingrandimento che apparirà sulla sua destra. Vi troverete di fronte ad una dialog box nella quale potrete inserire, premendo il pulsante +, nuovi elementi nell'array modificandone successivamente il testo con il nome effettivo che avete dato alle istanze di componenti. Una volta immessi tutti i nomi premete il tasto *ok* e correte a chiamare il capo… abbiamo terminato!! Non ci rimane infatti che testare l'applicazione pubblicando di nuovo il filmato come fatto in precedenza e quindi lanciando il file *.html* prodotto da FlashMX nella cartella dove risiede il programma. Appena effettuato il *Login* vi verrà chiesto se acconsentite all'utilizzo della webcam e del microfono e premendo il tasto *Consenti* potrete effettivamente vedere ciò che la vostra webcam stà riprendendo. Per fare una prova sulla multiutenza, potete far collegare un altro computer allo stesso file *.html*, verificando che Communication e FlashMX gestiscono perfettamente connessioni multiple e dando vita effettivamente alla videoconferenza preannunciata.

Fig. 4: La videoconferenza in azione!

Nel caso in cui non disponiate di una rete e di altri computers dotati di webcam, potete fare delle prove anche aprendo più volte sullo stesso computer la pagina html che lancia l'applicazione. Tenete presente che, avendo una sola webcam, le immagini che vedrete arrivare saranno ovviamente le stesse su entrambi i client connessi, e che FlashMX e Communication non gestiscono molto bene questa situazione dando vita spesso a latenze ingiustificate e connessioni instabili.

CONCLUSIONI

Visto come è facile, servendosi degli strumenti giusti, arrivare a gestire situazioni che avrebbero fatto perdere i capelli anche al più avvezzo dei programmatori? E non è finita qui, infatti vi sono molte possibili ulteriori impieghi del prodotto, che consente molto di più di quanto introdotto oggi, rendendo possibile l'impiego di Flash non solo come strumento per presentazioni internet di grande impatto. Restate sintonizzati, ne sentirete delle belle.

Giuliano Uboldi

Flash

Macromedia
Communication Server

SetBandWidth

Mette l'utente nella condizione di poter impostare la larghezza di banda con la quale si desidera ricevere le immagini degli altri utenti, rendendo possibile la diminuzione o l'aumento della qualità a seconda della connessione utilizzata.

Approfondimenti

Se vi interessa l'argomento potete collegarvi al sito www.macromedia.com/it/

e ricercare tra i prodotti Communication. Nel sito sono infatti presenti delle faq, dei tutorial e molto altro materiale sia per utenti alle prime armi che per programmatori esperti.

FOTOLAB

FILTRI FOTOGRAFICI IN VISUAL BASIC

PARTE SECONDA

Visual Basic

Dopo aver introdotto, nella prima parte di questo articolo, il formato BMP e le tecniche di trasformazione puntuale dell'immagine, in questa seconda ed ultima parte si affrontano le tecniche di trasformazione locale, che ci consentiranno di agire sul contrasto della stessa, di attenuarne il rumore elettronico ed anche di estrarre i contorni degli oggetti in essa rappresentati.

File sul CD

\soft\codice\filtri.zip

Il codice sorgente in Visual Basic 6 del programma FotoLab con l'implementazione dei suddetti algoritmi.

Ancora una volta gli algoritmi vengono sviluppati in Visual Basic, partendo da un file in formato BMP, limitandosi ad usare le librerie e i controlli standard di tale linguaggio; questo per consentire una agevole realizzazione degli stessi in altri linguaggi di programmazione.

L'OPERAZIONE DI CONVOLUZIONE

Le trasformazioni locali si basano sui pixel che si trovano nell'intorno di un determinato pixel, e consentono elaborazioni più sofisticate rispetto alle trasformazioni puntuali. Innanzitutto si deve definire l'intorno, o contorno, di un generico pixel: si possono fare scelte diverse, ma la più comune è quella di considerare tutti gli 8 pixel che gli stanno immediatamente attorno:

*	*	*
*		*
*	*	*

L'operazione di base con cui si rielabora il colore di un singolo pixel è la cosiddetta operazione di convoluzione che consiste nel calcolare l'intensità di ciascuno dei colori di base di un determinato pixel come combinazione lineare delle intensità dei corrispondenti colori dei pixel del suo intorno. I coefficienti di questa combinazione lineare vengono comodamente riportati in una matrice 3 x 3. Ad esempio, siano dati i seguenti livelli di rosso di un pixel (quello in neretto) e del suo intorno:

…	…	…	…	…
…	100	200	100	…
…	200	150	50	…
…	200	200	60	…
…	…	…	…	…

dove si ha $x(R) = 150$. Data la seguente matrice dei coefficienti di convoluzione:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

o, equivalentemente, la seguente matrice moltiplicata per il fattore 1/9:

1	1	1
1	1	1
1	1	1

si calcola il nuovo valore applicando i suddetti coefficienti ai corrispondenti pixel, e si ottiene:

$$y(R) = 1/9 * (100 + 200 + 100 + 200 + 150 + 50 + 200 + 200 + 60) = 140$$

In questo caso si tratta di una media aritmetica dei valori del pixel in esame e degli 8 pixel del suo intorno. I valori di y ottenuti da tale calcolo vengono ricondotti all'intervallo *0 .. 255*, assegnando il valore *0* agli *y* negativi e il valore *255* agli *y* maggiori di *255*. Si nota che non è possibile applicare tale operazione ai pixel del bordo dell'immagine, pertanto essi non saranno sottoposti a tale trasformazione.

TRASFORMAZIONI LOCALI

Con diverse matrici di convoluzione si possono effettuare diverse interessanti trasformazioni dell'immagine, tra cui quelle di seguito schematicamente riportate.

1) filtro **Media Semplice**

effetti: sfuocamento dell'immagine (blurring)
fattore moltiplicativo: 1/9
matrice di convoluzione:

1	1	1
1	1	1
1	1	1

una media aritmetica semplice delle intensità dei colori dei pixel dell'intorno del pixel considerato, pertanto è intuitivo aspettarsi una specie di dissolvimento localizzato dei dettagli dell'immagine.

2) filtro **Media Gaussiana**, o **Attenuatore**
effetti: lisciamento e attenuazione del rumore e del contrasto (softening)
fattore moltiplicativo: 1/16
matrice di convoluzione:

1	2	1
2	4	2
1	2	1

una media aritmetica ponderata delle intensità dei colori dei pixel dell'intorno del pixel considerato dove si da peso maggiore al pixel centrale e peso via via decrescente ai pixel che gli stanno attorno.

3) filtro **Laplaciano Modificato**, o **"Sharp Enhancer"**
effetti: aumenta il contrasto (sharpening), con l'effetto indesiderato di introdurre del rumore
fattore moltiplicativo: 1
matrice di convoluzione:

0	-1	0
-1	5	-1
0	-1	0

un operatore che sostanzialmente moltiplica per 5 la differenza di intensità di colore del pixel centrale rispetto alla media dei quattro pixel immediatamente contigui; in questo modo si aumenta localmente il contrasto dell'immagine.

4) filtro **Laplaciano Potenziato**
effetti: evidenziazione dei contorni (*edge detection*)
fattore moltiplicativo: 1

1	1	1
1	-9	1
1	1	1

una versione più potente dell'operatore laplaciano che aumenta a tal punto il contrasto che i bordi diventano bianchi e la restante parte dell'immagine diventa nera, grazie anche ad un opportuno cambio di segno. Per esaltare ulteriormente i contorni, è opportuno effettuare anche la trasformazione dell'immagine in livelli di grigio, parificando i livelli delle intensità di colore al valore massimo assunto dai tre colori di base.

5) filtro **Mediana**
effetti: lisciamento e attenuazione del rumore (smoothing), in particolare di tipo sale e pepe

una trasformazione locale che si distingue dalle precedenti perché non si basa sull'operazione di convoluzione.
Essa consiste nell'assegnare a ciascun pixel il livello mediano delle intensità di colore degli 8 pixel del suo intorno unitamente al pixel considerato. In questo modo si ottiene un effetto analogo a quelli della media semplice e della media gaussiana, con la differenza però che il risultato non viene influenzato da singoli pixel troppo chiari o troppo scuri, che sono presumibilmente l'effetto di distorsioni nell'acquisizione dell'immagine.
Ecco quindi l'eliminazione del rumore cosiddetto "sale e pepe", che appunto è costituito dalla presenza casuale nell'immagine di punti eccessivamente chiari o scuri.

Esempio

A titolo esemplificativo si mostrano gli effetti delle suddette elaborazioni sull'immagine di Fig. 1:

Fig. 1: **L'immagine di partenza.**

Dapprima si applica il filtro media semplice e si ottiene l'immagine sfocata di Fig. 2a. Seguono gli effetti di leggera attenuazione del contrasto del filtro media gaussiana e di attenuazione del rumore di tipo sale e pepe del filtro mediana (Fig. 2b e 2c). In effetti l'immagine di partenza non era particolarmente soggetta a rumore e quindi non si percepisce molto l'effetto di quest'ultimo filtro. Poi si applica, sempre all'immagine di partenza, il filtro

Fig. 2a: **Immagine Sfuocata.** **Fig. 2b:** **Attenuazione del Contrasto.** **Fig. 2c:** **Attenuazione del Rumore.**

Filtro Mediana

Il filtro Mediana calcolato considerando gli 8 pixel attorno al pixel in esame ha lo svantaggio di danneggiare le linee sottili e gli spigoli vivi presenti nell'immagine. Per evitare questi inconvenienti si può utilizzare il filtro Mediana che preserva le linee orizzontali e verticali, che si calcola effettuando la mediana delle intensità dei colori dei pixel del seguente intorno del pixel considerato:

		*		
		*		
*	*		*	*
		*		
		*		

Visual Basic

Fotolab
Filtri Fotografici in Visual Basic

“sharp enhancer” per aumentarne il contrasto (Fig. 2d). Infine viene applicato il fitro evidenziatore dei contorni (Fig. 2e) e, per vederne meglio l’effetto, si prende il suo risultato e se ne fa il negativo (Fig. 2f).

Fig. 2d: Aumento del Contrasto.
Fig. 2e: Evidenziazione dei Contorni.
Fig. 2f: Evidenziazione dei Contorni in Negativo.

CODIFICA IN VISUAL BASIC

La Fig. 3 mostra la finestra del programma FOTOLAB. Per comodità espositiva si ricorda la struttura dati utilizzata da tale programma.

Fig. 3: FotoLab.

A livello globale viene dichiarato un array di oggetti “foto”, ciascuno dei quali avrà il nome della foto, il riferimento al Form su cui essa viene visualizzata, le dimensioni della stessa, l’offset dei pixel e gli array di byte contenenti l’intestazione del file e la matrice di pixel.

```
Private Type tipoFoto
   NomeFoto As String
   FormFoto As frmFoto
   w As Long   ' ampiezza foto
   w3 As Long  ' ampiezza aggiustata
   h As Long   ' altezza foto
   inizioPixel As Long  ' offset
   header() As Byte  ' header del file BMP
   g() As Byte   ' matrice di pixel
End Type

Dim Foto() As tipoFoto   ' array di foto
Dim nFoto As Integer ' numero di foto caricate nell’array
Dim gCorrente As Integer  ' indice della foto corrente
```

La foto ha dimensione $w * h$ pixel, quindi essa occuperà una matrice di Byte con h righe e $w3 = w * 3$ colonne (dove in effetti $w3$ viene arrotondato al successivo multiplo di 4 a causa delle specifiche del formato BMP). Si ricorda che la matrice g dei pixel contiene per ciascun pixel tre componenti per memorizzare le intensità dei tre colori di base *Red*, *Green* e *Blue:*

	1	2	3	4	5	6	...	w*3
1	Blue 1	Green 1	Red 1	Blue 2	Green 2	Red 2	...	...
...	...	...	...	...	...	...	...	...
h	...	...	...	...	...	...	...	...

Per l’applicazione dei filtri locali, che richiedono l’operazione di convoluzione, si veda a titolo esemplificativo la seguente subroutine che aumenta il contrasto dell’immagine applicando il filtro *Sharp Enhancer*. Dapprima vengono impostati la matrice di convoluzione e il fattore moltiplicativo e poi, dopo aver creato un duplicato della foto da elaborare, si chiama la subroutine che effettua la trasformazione desiderata.

```
Private Sub mnuContrasta_Click()
Dim Origine As Integer
Dim NomeFoto As String
Dim mat(1 To 3, 1 To 3) As Integer
Dim fattore As Single
' matrice di convoluzione
mat(1, 1) = 0
mat(1, 2) = -1
mat(1, 3) = 0
mat(2, 1) = -1
mat(2, 2) = 5
mat(2, 3) = -1
mat(3, 1) = 0
mat(3, 2) = -1
mat(3, 3) = 0
' fattore moltiplicativo
fattore = 1
' aggiunge una nuova componente all’array delle foto
Origine = gCorrente
nFoto = nFoto + 1
ReDim Preserve Foto(1 To nFoto)
gCorrente = nFoto
NomeFoto = "BMP_" & CStr(nFoto)
' carica la foto nell’array  delle foto
CaricaFoto NomeFoto, Origine
' chiamata della routine di elaborazione vera e propria
' passando come parametri la matrice di convoluzione
' e il fattore moltiplicativo
FiltraConvFoto mat, fattore
End Sub
```

Ecco ora la subroutine *FiltraConvFoto*, che agisce sulla foto corrente, che è quella indicata dall’indice *gCorrente*. Mediante due cicli nidificati, si scandisce la matrice dei pixel e per ciascuna componente di colore di ciascun pixel si applica la funzione di convoluzione.
Si ricorda che gli indici di riga e di colonna della matrice risultano invertiti rispetto alle usuali convenzioni a causa del meccanismo utilizzato per

Applicazione

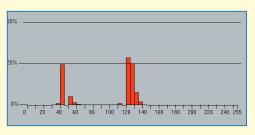
Una innovativa applicazione delle tecniche di elaborazione di immagini è la realizzazione di sistemi di visione utilizzati per misurare le dimensioni di oggetti prodotti da un processo di produzione, con lo scopo di effettuare un controllo automatico dello stesso.
Si faccia riferimento al libro del prof. Giulio Fanti del Dipartimento di Ingegneria Meccanica dell’Università di Padova, intitolato

- **SISTEMI DI VISIONE PER MISURE DIMENSIONALI**
Giulio Fanti
(Ed. Libreria Progetto) 1999

Visual Basic

Fotolab
Filtri Fotografici in Visual Basic

L'istogramma dei livelli di grigio

L'istogramma di una immagine a livelli di grigio è un modo per caratterizzare l'immagine stessa, rappresentando per ognuno dei 256 livelli di grigio il numero di pixel dell'immagine che assumono tale livello.

Si tratta in sostanza di un diagramma delle frequenze relative dei livelli di grigio.

l'acquisizione della stessa dal file BMP.

```
Private Sub FiltraConvFoto(ByRef mat() As Integer,
                                        ByVal fattore As Single)
Dim i As Long
Dim j As Long
Dim n As Long
Dim g2() As Byte
' ovviamente sono escluse dall'elaborazione la prima e
' l'ultima riga e la prima e l'ultima colonna della matrice
With Foto(gCorrente)
  ' calcolo dove termina il penultimo pixel di ciascuna riga
  n = 3 * (.w - 1)
  ' duplico la matrice di partenza
  g2 = .g
For i = 2 To .h – 1
    j = 4   ' parto dal secondo pixel della riga: 3 + 1 = 4
    Do While j <= n
       g2(j, i) = convoluzione(i, j, .g, mat, fattore)
       j = j + 1
    Loop
  Next i
  ' assegno il risultato alla matrice di partenza
  .g = g2
  ' salvo la foto in un file temporaneo
  ' e poi la visualizzo
  ...
End With

End Sub
```

Ecco, infine, la *Function convoluzione()*, che viene calcolata per ciascun singolo colore di ciascun pixel.
Essa effettua il calcolo sulla componenente corrente della matrice *g* dei pixel, indicata dalle coordinate riga e colonna, e utilizza la matrice di convoluzione e il fattore moltiplicativo ricevuti come parametri.

```
Private Function convoluzione(ByVal riga As Integer,
      ByVal colonna As Integer, ByRef g() As Byte, ByRef
   matrice() As Integer, ByVal fattore As Single) As Byte
Dim a As Integer, b As Integer
Dim i As Integer, j As Integer
Dim r As Single   ' il risultato della convoluzione
r = 0
i = 0
For a = riga - 1 To riga + 1
  i = i + 1
  j = 0
  b = colonna - 3
  Do While b <= colonna + 3
    j = j + 1
    r = r + g(b, a) * matrice(i, j)
    b = b + 3
  Loop
Next a
r = r * fattore

' si riconduce il risultato all'intervallo 0 .. 255
If r < 0 Then
  r = 0
ElseIf r > 255 Then
  r = 255
End If

' si assegna il valore di ritorno convertito in Byte
convoluzione = CByte(r)
End Function
```

CONCLUSIONI

Si conclude questa breve introduzione alle problematiche legate al trattamento delle immagini bitmap, dove sono state affrontate le trasformazioni di base che costituiscono il corredo essenziale di qualsiasi programma di ritocco fotografico.

Roberto Bandiera

Compressione di immagini

Le immagini a 24 bit di profondità di colore memorizzate in formato BMP occupano una notevole quantità di spazio su disco, pertanto risulta conveniente comprimerle secondo il formato JPG, in modo da ridurne notevolmente le dimensioni.
Ad esempio una foto BMP a 24 bit da 1024x768 pixel occupa 2,25 MB mentre la corrispondente foto compressa in formato JPG occupa tipicamente poco più di 100 KB. La compressione in formato JPG comporta una alterazione dell'immagine di partenza, che viene a perdere alcune sfumature di colore, ma si tratta di una modifica poco percettibile in quanto l'uomo non è in grado di distinguere tutti i 16 milioni di colori disponibili con la codifica RGB a 24 bit ma soltanto un numero di colori che si aggira attorno ai 350.000.

Riferimenti

Per approfondimenti e una ampia rassegna di tecniche di elaborazione di immagini si visiti il sito del Dipartimento di Intelligenza Artificiale dell'Università di Edinburgo
http://www.dai.ed.ac.uk/HIPR2/hipr_top.htm

Presso il Dipartimento di Computer Science della stessa università sono reperibili informazioni sui formati grafici
http://www.dcs.ed.ac.uk/home/mxr/gfx/2d-hi.html

Per una introduzione alla fotografia digitale si consulti il manuale on line "Le Tecniche Fotografiche In Archeologia" di Fausto Gabrielli del Dipartimento di Scienze Archeologiche – Università di Pisa
http://www.arch.unipi.it/Foto_Libro/Foto_Libro.html

Soap

Accedere

A DATABASE REMOTI CON WEB SERVICES .NET

Il mese scorso abbiamo visto come sia possibile costruire, in modo piuttosto semplice, un Web Service utilizzando la piattaforma Microsoft .Net ed in particolare il Microsoft .Net Framework SDK.

File sul CD
\soft\codice\Soap.zip

Abbiamo visto anche come un semplice file sorgente VB.Net o C#, con le opportune personalizzazioni, sia immediatamente disponibile per essere pubblicato sotto forma di Web Service e come la piattaforma .Net permetta di avere in modo gratuito un client HTTP per utilizzare da subito il Web Service appena costruito. In questo articolo, per fare un passo in avanti cercheremo di capire come sfruttare al meglio alcune caratteristiche dell'ambiente e della piattaforma Microsoft .Net. Per fare questo svilupperemo un Web Service più completo, in grado di trattare tipi primitivi ma anche oggetti complessi, con l'obiettivo di fornire le basi per un sistema di profilatura utente distribuibile attraverso il protocollo SOAP come Web Service. Più in generale svilupperemo un metodo per accedere da un client al database remoto della profilatura.

SCENARIO

Immaginiamo di avere due reti Intranet disgiunte ed entrambe connesse alla rete Internet. Ogni rete locale ha un proprio proxy/firewall che isola tutte le macchine della rete interna dalla rete Internet garantendo al tempo stesso connettività, sicurezza ed IP masking. Uno dei meccanismi cardine della sicurezza consiste nel bloccare il traffico agendo sulle porte TCP/IP, impedendo quindi, agli utenti delle reti interne, l'utilizzo di porte non convenzionali. In particolare risulterà sempre aperta, a parte casi particolari, la porta 80, propria del protocollo HTTP. Immaginiamo di avere, su una delle reti Intranet, un servizio di autenticazione e profilatura utente trasversale che serva le applicazioni della rete stessa, e di volerlo estendere per renderlo fruibile anche dall'esterno, in particolare dall'altra rete Intranet. In uno scenario come questo sembra molto difficile far comunicare applicativamente tra loro le due reti in quanto il protocollo HTTP è dedicato esclusivamente, per sua natura, al traffico Web. Grazie al protocollo SOAP, però, sappiamo che è possibile far comunicare due oggetti software attraverso un protocollo di trasporto in grado di muovere del semplice testo (ed in questo caso HTTP va benissimo) ed attraverso la serializzazione in XML del pacchetto informativo che le due componenti software si devono scambiare. Grazie a SOAP ed utilizzando la piattaforma Microsoft .Net proveremo a risolvere il problema attraverso lo sviluppo di un Web Service.

IL SERVIZIO

Quello che costruiremo sarà un servizio di profilatura utente scritto in C# il cui nome sarà *Profiling*. In particolare, trattandosi di un Web Service .Net, il file che implementa il servizio prende il nome di *profiling .asmx*. Le funzionalità messe a disposizione dal servizio sono identificate attraverso lo *Use Case Diagram* mostrato in Fig. 1.

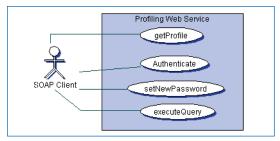



Fig. 1: **Le funzionalità esposte dal Web Service.**

Il requisito iniziale prevede quindi quattro funzionalità distinte che dovranno essere messe a disposizione attraverso il servizio Web. L'implementazione del Web Service dovrà fornire il risultato mostrato in Fig. 2. Analizzando il codice del servizio è possibile notare immediatamente qualche differenza rispetto all'implementazione del Web Service del mese scorso, in particolare si tratta dei seguenti frammenti di codice:

```
[WebService(
   Name="Profiling",
   Description="Applicazione di gestione della
                              profilatura aziendale",
   Namespace="max.profiling")]
public class Profiling : WebService
{ … }
```

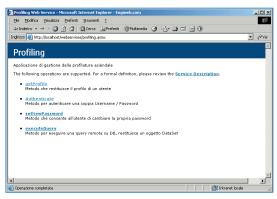



Fig. 2: La pagina di accesso al Web Service.

Campo	Tipo
ID	contatore;
Username	testo(50);
Password	testo(50);
Nome	testo(50);
Cognome	testo(50);
Indirizzo	testo(50);
Email	testo(50);
Struttura	testo(50)

Tab. 1: Struttura della tabella db.

Soap

Accedere

a database remoti con Web Services .Net

In questo caso sono stati utilizzati alcuni attributi del modificatore *[WebService]* per identificare il nome del servizio, la sua descrizione ed il *NameSpace* di riferimento. L'effetto di questi attributi è subito chiaro guardando la Fig. 2, dove si vede che il nome del servizio e la sua descrizione compaiono nella pagina di accesso al servizio stesso, inoltre l'aver indicato la presenza di un *NameSpace* ha fatto sparire il warning che segnalava l'assenza di un *NameSpace* specifico.
La pagina di accesso al servizio appare molto più pulita ed ordinata, anche grazie alle descrizioni dei singoli metodi che il servizio espone ai consumatori. Questo avviene grazie alla presenza dell'attributo *Description* sul modificatore dei singoli metodi:

```
[WebMethod(Description="Metodo per eseguire una
  query remota su DB, restituisce un oggetto DataSet")]
public DataSet executeQuery(string Query)
{ …}
[WebMethod(Description="Metodo per autenticare una
                        coppia Username / Password")]
public bool Authenticate(string Username, string Password)
{ …}
[WebMethod(Description="Metodo che restituisce il
                              profilo di un utente")]
public DataSet getProfile(string Username, string Password)
{ …}
[WebMethod(Description="Metodo che consente
          all'utente di cambiare la propria password")]
public string setNewPassword(string Username,
              string OldPassword, string NewPassword)
{ …}
```

IMPLEMENTAZIONE

L'intero codice del servizio, e tutti gli altri file descritti in questo articolo, sono disponibili sul CD allegato alla rivista. La base dati è contenuta in un database Access dal nome *profili.mdb*. Il database è composto dall'unica tabella *Utenti* che ha la struttura di Tab. 1.
Dovendo utilizzare una connessione a database è necessario utilizzare un *NameSpace* specifico, si tratta di *System.Data.OleDb*. Inoltre sul sistema, per poter accedere ad un database SQL oppure OleDb, deve essere installato il Microsoft Data Access Component 2.6 reperibile all'indirizzo: *http://download.microsoft.com /download/MDAC26/SP/1.0/W98NT42KMeXP/IT/MD AC_TYP.EXE* Altra peculiarità di questo servizio è la presenza, all'interno del codice dello stesso, di un metodo che, a differenza degli altri, non viene esposto come *WebMethod*, ma è invece un metodo privato che può essere utilizzato solo dagli altri metodi implementati all'interno della stessa classe. Si tratta del metodo *isValidLogin* che prende in input le stringhe *Username* e *Password* e restituisce *true* o *false* in funzione della validità o meno dei parametri. In particolare restituisce *true* se esiste nella tabella *Utenti* un record con *Username* e *Password* conformi ai parametri ricevuti.

```
// restituisce true se la coppia Username/Password
// è valida, in caso contrario ritorna false
private bool isValidLogin(string Username, string Password)
{ …}
```

Seguendo la filosofia OOP, ci interessa "cosa fa" un servizio, e nel caso specifico cosa fanno i metodi esposti dal servizio stesso e non "come" questi sono implementati. Per analizzare il dettaglio dell'implementazione dei singoli metodi del servizio vi rimando ai file allegati alla rivista.

INSTALLAZIONE

Se avete letto l'articolo del mese scorso dovreste già avere la directory virtuale *WebServices* sul vostro Internet Information Services. Directory cui saranno stati assegnati i permessi di esecuzione di script. Per rendere operativo il Web Service è necessario copiare il file *profiling.asmx* ed il database cui si appoggia *Profili.mdb* all'interno del percorso fisico che viene mappato come directory virtuale WebServices da IIS. Poiché ci si aspetta che i dati sul database possano variare, è necessario che il percorso fisico su cui è mappata la directory virtuale sia modificabile dall'utente con cui parte il servizio di IIS. In caso contrario il processo di esecuzione degli script ASP.Net non potrà apportare modifiche al database rendendo parzialmente non operativo il nostro Web Service. Questa configurazio-

WSDL

WSDL è una specifica del W3C http://www.w3c.org/TR/wsdl che ha l'obiettivo di fornire la descrizione di un Web Service utilizzando un'applicazione di XML. Poiché i Web Services sono per definizione distribuiti su piattaforme differenti e consumati da client che a priori non sono noti, è bene definire un file WSDL per ogni servizio che si produce in modo da disaccoppiare il servizio dal client che cerca di utilizzarlo. In questo modo il client non utilizza direttamente il Web Service, ma passa attraverso la sua descrizione in WSDL per capire come fare ad utilizzare il servizio.

Soap

Accedere
a database remoti con Web Services .Net

ne può variare molto tra un'installazione e l'altra, ma un metodo sicuro per raggiungere il risultato consiste nell'assegnare all'utente *Everyone* il controllo completo sulla posizione fisica mappata dalla directory virtuale. Dopo aver seguito tutti i passi dell'installazione e dopo aver fatto partire il servizio di Internet Information Services, il nostro nuovo Web Service sarà raggiungibile direttamente dall'url: *http://localhost/webservices/profiling.asmx* ed il risultato sarà quello visibile in Fig. 2.

USARE IL SERVIZIO AUTHENTICATE

Il Web Service che abbiamo installato espone quattro metodi, ognuno dei quali può essere utilizzato direttamente dal browser oppure attraverso un client scritto ad hoc. Proviamo ad esempio a vedere il funzionamento del metodo *Authenticate*, una volta effettuato il click sul nome del metodo ci vengono richiesti (Fig. 3) i valori da assegnare ai due parametri che il metodo remoto si aspetta: *Username* e *Password*.

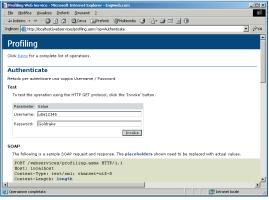



Fig. 3: L'interazione con il Web Service, attraverso il metodo ***authenticate***.

Inseriamo *u0e12345* come valore per il campo *Username* e *Goldrake* come valore per il campo Password. Dopo la pressione del pulsante *Invoke* il servizio remoto verrà eseguito e ci restituirà un payload XML contenente il valore true in quanto sul nostro database esiste un record per quell'utente che ha esattamente quella password. Il risultato è visibile in Fig. 4. Naturalmente se avessimo inserito una password sbagliata il metodo remoto ci avrebbe risposto con lo

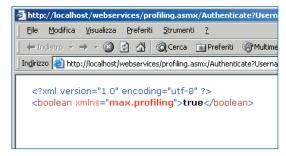



Fig. 4: Pagina restituita dalla pressione del pulsante ***Invoke.***

stesso payload XML, ma questa volta contenente il valore false in quanto l'autenticazione sarebbe fallita.

USARE IL SERVIZIO GETPROFILE

Il discorso è analogo se si utilizza il metodo *getProfile*, anch'esso riceve come parametri la coppia *Username* e *Password*, ma questa volta, a fronte di un riscontro positivo dell'autenticazione, il metodo restituisce un oggetto *DataSet* che, in questo caso, contiene un solo record. Una porzione del payload restituita dal metodo, è visibile in Fig. 5.

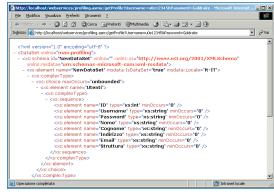



Fig. 5: Il payload restituito da ***GetProfile.***

Analizzando questo payload XML si può notare che è composto da due parti, la prima sezione è lo schema applicato che viene derivato da elementi contenuti nel NameSpace XML *msdata="urn:schemas-microsoft-com:xml-msdata*". Le seconda sezione, invece , contiene i dati veri e propri nella forma indicata dallo schema della sezione precedente. Questa distinzione in sezioni è fondamentale in quanto un oggetto *DataSet* ha una componente strutturale (che contiene il nome della tabella e le informazioni su nome e tipo delle singole colonne), ed una componente dati (che contiene tutti i singoli record impacchettati secondo la struttura precedente). Una cosa da notare è che, mentre con l'utilizzo del metodo *Authenticate* viene garantita l'interoperabilità in quanto il valore restituito è il tipo primitivo booleano, utilizzando il metodo *getProfile*, il valore di ritorno appartiene ad un tipo che potrebbe non essere riconosciuto da un consumatore sviluppato con tecnologia differente. In particolare è naturale pensare che un client sviluppato in C# su piattaforma .Net sia in grado di utilizzare un oggetto *DataSet*, è più difficile invece ottenere la comprensione e l'effettivo utilizzo del tipo *DataSet* da parte di un client scritto, per esempio, con tecnologia Java.

USARE IL SERVIZIO SETNEWPASSWORD

Vediamo a questo punto l'utilizzo del metodo *setNewPassword*. In Fig. 6 si può vedere la descrizione del metodo ed il form per utilizzarlo, questa pagina è

UDDI

Spesso avere un grande repository di informazioni o di servizi non è sufficiente, è necessario anche avere gli strumenti per trovare l'informazione o il servizio giusto quando serve. Mentre sul Web questo è il compito dei motori di ricerca, nel panorama dei Web Services è stato costituito, da un consorzio di più di 300 aziende, un database in grado di immagazzinare tutti i servizi web offerti dalla aziende stesse con l'obiettivo poter ricercare i servizi e le informazioni dettagliate per accedere ai servizi stessi.
La specifica di disegno ed utilizzo di queste informazioni è UDDI (Universal Description Discovery and Integration).

Soap

Accedere
a database remoti con Web Services .Net

Fig. 6: **Utilizzo del metodo SetNewPassword.**

stata ottenuta dopo aver effettuato un click sul nome del metodo stesso dalla pagina di descrizione del servizio. Il metodo *setNewPassword* che viene esposto dal servizio è implementato in questo modo:

```
public string setNewPassword(string Username,
                    string OldPassword, string NewPassword)
{ try
  { if (isValidLogin(Username, OldPassword))
    { // qua viene fatta l'operazione di UPDATE
      return "ok";
    }
     else return "invalid login";
   }
   catch(Exception ex)
   {
      return "error: " + ex.Message;
   }
}
```

Per testarlo possiamo inserire i valori *u0e12345*, *Goldrake* e *Mazinga* rispettivamente per i parametri *Username*, *OldPassword* e *NewPassword*. Se i dati sul database non sono stati preventivamente modificati, la password dell'utente indicato sarà modificata attraverso l'operazione di *UPDATE* ed il metodo restituirà il seguente payload XML:

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="max.profiling">ok</string>
```

Se provassimo ad eseguire nuovamente lo stesso metodo con gli stessi parametri otterremmo invece il seguente payload XML:

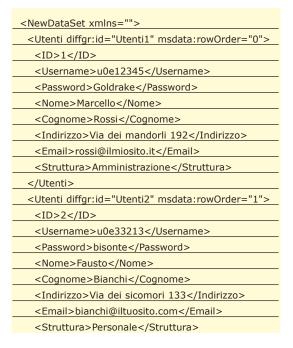
```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="max.profiling">invalid login</string>
```

in quanto la password dell'utente è già stata cambiata dalla prima esecuzione del metodo e quindi un tentativo di autenticazione con la vecchia password non andrà a buon fine.
In caso di errori differenti, ad esempio se il sistema non trova il database o se vi siete dimenticati di dare all'utente *Everyone* i permessi di controllo completo sulla directory che contiene il database, il payload conterrà un messaggio di errore.

USARE IL SERVIZIO EXECUTEQUERY

Il servizio che abbiamo sviluppato espone anche il metodo *executeQuery*. Tale metodo prende in input una qualsiasi query SQL (compatibile con il linguaggio SQL utilizzato dal provider che garantisce l'accesso al database) e restituisce un oggetto di tipo *DataSet* contenente il risultato dell'elaborazione della query sul database stesso. Per testare il servizio utilizziamo nuovamente la pagina di accesso al metodo (visibile in Fig. 7) ed inseriamo la query *select * from Utenti*. Premendo il pulsante *Invoke* otterremo un payload XML formato nuovamente da due sezioni, la prima è, come accadeva per il metodo *getProfile*, la struttura del *DataSet*, la seconda invece è composta da tutti i record del *DataSet* ritornato dal metodo, vediamo questa seconda parte del payload:

Fig. 7: **Interrogazione del database attraverso il metodo ExecuteQuery.**

```
<NewDataSet xmlns="">
  <Utenti diffgr:id="Utenti1" msdata:rowOrder="0">
    <ID>1</ID>
    <Username>u0e12345</Username>
    <Password>Goldrake</Password>
    <Nome>Marcello</Nome>
    <Cognome>Rossi</Cognome>
    <Indirizzo>Via dei mandorli 192</Indirizzo>
    <Email>rossi@ilmiosito.it</Email>
    <Struttura>Amministrazione</Struttura>
  </Utenti>
  <Utenti diffgr:id="Utenti2" msdata:rowOrder="1">
    <ID>2</ID>
    <Username>u0e33213</Username>
    <Password>bisonte</Password>
    <Nome>Fausto</Nome>
    <Cognome>Bianchi</Cognome>
    <Indirizzo>Via dei sicomori 133</Indirizzo>
    <Email>bianchi@iltuosito.com</Email>
    <Struttura>Personale</Struttura>
```

SOAP

SOAP è un protocollo applicativo che fornisce le regole per permettere ad applicazioni distribuite di scambiarsi informazioni e di richiedere l'esecuzione di servizi remoti. Le applicazioni si scambiano messaggi attraverso un pacchetto informativo che prende il nome di payload, una struttura XML complessa. Non esistono restrizioni per quel che riguarda il protocollo di trasporto utilizzato, l'unico vincolo è che sia in grado di trasportare il semplice testo.
In realtà, sebbene non esistano controindicazioni di sorta verso altro protocolli, si tende a privilegiare HTTP per la sua ampia distribuzione e per le caratteristiche di request/response simili a quelle di SOAP.

Soap

Accedere
a database remoti con Web Services .Net

Requisiti minimi

Per utilizzare il Microsoft .Net Framework SDK per lo sviluppo di Web Service o applicazioni ASP.Net sono necessari i seguenti requisiti minimi:

HARDWARE:
- ***Processore:* Intel Pentium a 133 MHz o superiore**
- ***RAM:* 128 MB (256MB consigliati)**
- ***Spazio disponibile su disco rigido necessario per l'installazione:* 600 MB**
- ***Spazio disponibile su disco rigido necessario:* 370 MB**
- ***Monitor:* 800x600, 256 colori.**

SISTEMA OPERATIVO:
- **Microsoft Windows 2000 SP2 o superiore.**
- **Microsoft Windows XP Professional e Home.**
- **Microsoft Windows NT 4.0 con Service Pack 6a o versione successiva.**

ALTRO SOFTWARE:
- **Internet Information Services (per lo sviluppo di applicazioni ASP.Net o Web Services).**
- **Microsoft Data Access Component 2.6 (per l'accesso ai dati).**

```
  </Utenti>
  <Utenti diffgr:id="Utenti3" msdata:rowOrder="2">
    <ID>3</ID>
    <Username>u0e11223</Username>
    <Password>casdckjflkejbfekj</Password>
    <Nome>Maria</Nome>
    <Cognome>Furbetta</Cognome>
    <Indirizzo>Corso del tempo 1</Indirizzo>
    <Email>maria@abracadabra.it</Email>
    <Struttura>Uffici periferici</Struttura>
  </Utenti>
</NewDataSet>
```

Ecco quindi tutti i record generati dall'esecuzione remota della query sul database. Proviamo una query più puntuale inserendo, nel form contenuto nella pagina di Fig. 7, la query:

```
select indirizzo from Utenti where username='u0e12345'
```

Eseguiamo il metodo remoto ed il payload che otterremo sarà il seguente:

```
<?xml version="1.0" encoding="utf-8"?>
<DataSet xmlns="max.profiling">
  <xs:schema
    id="NewDataSet"
    xmlns=""
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xs:element name="NewDataSet"
        msdata:IsDataSet="true" msdata:Locale="it-IT">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="Utenti">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="indirizzo"
                        type="xs:string" minOccurs="0" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:schema>
  <diffgr:diffgram
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
    xmlns:diffgr="urn:schemas-microsoft-com:
                                xml-diffgram-v1">
    <NewDataSet xmlns="">
      <Utenti diffgr:id="Utenti1" msdata:rowOrder="0">
        <indirizzo>Via dei mandorli 192</indirizzo>
      </Utenti>
    </NewDataSet>
  </diffgr:diffgram>
</DataSet>
```

In questo caso il *DataSet* restituito dal metodo remoto conterrà soltanto il campo *"Indirizzo"* identificato dallo schema come *type="xs:string"* esattamente come indicato nella struttura del database. Utilizzando questo metodo è possibile anche lanciare query di modifica sul database remoto, proviamo ad esempio ad invocare il metodo con la query:

```
UPDATE Utenti SET Utenti.Email = 'nuovoindirizzo@email.it'
                      where Utenti.Username='u0e12345'
```

In questo caso il payload restituito sarà un *DataSet* vuoto contenuto nel payload seguente:

```
<?xml version="1.0" encoding="utf-8"?>
<DataSet xmlns="max.profiling">
  <xs:schema
    id="NewDataSet"
    xmlns=""
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:msdata="urn:schemas-microsoft-com:
                                  xml-msdata">
    <xs:element name="NewDataSet" msdata:
            IsDataSet="true" msdata:Locale="it-IT">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded" />
      </xs:complexType>
    </xs:element>
  </xs:schema>
  <diffgr:diffgram
    xmlns:msdata="urn:schemas-microsoft-com:
                                  xml-msdata"
    xmlns:diffgr="urn:schemas-microsoft-com:
                          xml-diffgram-v1" />
</DataSet>
```

Ma andando a verificare il contenuto del database possiamo notare che la query di aggiornamento è stata eseguita senza problemi.

CONCLUSIONI

Abbiamo visto come scrivere un servizio di profilatura remota implementato come Web Service in tecnologia Microsoft .Net. I metodi remoti messi a disposizione dal Web Service implementano alcune delle funzioni tipiche della profilatura utenti, come l'autenticazione, il recupero del proprio profilo o il cambio della password. A questi metodi è stato aggiunto il più generale *executeQuery* che permette di effettuare qualunque query sul database remoto gestendo lo scambio dei dati attraverso un payload XML, in particolare il client ed il server si scambiano dei veri e propri oggetti *DataSet*. Avendo quindi a disposizione un client in grado di utilizzare il tipo *DataSet*, sarà possibile eseguire qualsiasi query sul database remoto e gestirla applicativamente come se la query fosse eseguita su un database locale, questo sarà l'argomento del prossimo articolo.

Massimo Canducci

VoiceXML

XML FA SENTIRE LA SUA VOCE

Il World Wide Web Consortium ha reso pubbliche le nuove specifiche del linguaggio standard per la gestione di dialoghi audio uomo-macchina. In questo articolo esamineremo la sintassi di base, i possibili impieghi ed i prodotti concorrenti della lingua franca per la telefonia: VoiceXML.

Uno dei compiti di una rivista come ioProgrammo è quello di proporre all'attenzione del lettore tecnologie innovative che difficilmente trovano spazio su libri o altri media. L'argomento trattato in questo articolo è quanto mai attuale perché risulta collegato a temi "caldi" come: telefonia mobile, domotica, wireless e interazione multimodale.

INTRODUZIONE

VoiceMXL è un linguaggio basato su XML, giunto alla versione 2.0, ideato appositamente per agevolare la creazione di interfacce utente vocali. Immagino che in questi termini sia difficile percepirne l'effettiva utilità, vediamo dunque in quali contesti viene prevalentemente impiegato:

- **Interactive Voice Response, in breve IVR:** i sistemi IVR consentono l'interazione vocale, di solito tramite telefono, tra un utente ed un centro informazioni. Gli esempi più classici sono: il "servizio clienti" dei gestori di telefonia mobile (non avete mai chiamato il 190 o numeri simili?), caselle postali e portali vocali, chat, prenotazione di biglietti, chioschi informativi, etc.

- **Domotica:** è la scienza che si occupa dell'integrazione di informatica ed elettronica nelle abitazioni. Probabilmente pur conoscendo l'espressione casa intelligente ignorate che sia possibile impartire comandi alla propria abitazione del tipo "Spegni la luce del soggiorno" e sentirsi rispondere da un maggiordomo virtuale "Luce soggiorno spenta". Non è fantascienza, ma per pochi fortunati la realtà quotidiana!

- **Navigazione Web:** VoiceXML sta ai browser vocali come HTML sta ai più noti web-browser. Oltre a rappresentare un modo nuovo di concepire la navigazione Web, a dire la verità in generale non molto comodo, VoiceXML garantisce una maggiore accessibilità a chi ha problemi di vista o nell'utilizzo di dispositivi quali tastiera e mouse. E' anche possibile creare senza troppe difficoltà delle Intranet per lo scambio di informazioni audio/video.

- **Applicazioni telefoniche:** molte operazioni vengono notevolmente semplificate da un'interfaccia vocale, è sufficiente pensare all'interrogazione di database o all'uso di un'agenda telefonica capace di ricordare appuntamenti, compleanni o scadenze con voce suadente.

Ovviamente ho proposto solo qualche campo applicativo del linguaggio VoiceXML, in realtà ogni volta che l'uso della voce può migliorare un prodotto in maniera sostanziale è bene valutare se conviene puntare su tale standard. Prima di passare all'analisi dell'architettura e della sintassi vi faccio notare che in una tipica applicazione VoiceXML l'input dei dati avviene tramite riconoscimento vocale o sequenze DTMF mentre per l'output si usa la sintesi text-to-speech (TTS, il testo viene letto da un apposito software) o semplici messaggi preregistrati. Siamo ancora lontani dai livelli di HAL9000, il supercomputer di 2001 A Space Odyssey, ma è soltanto una questione di tempo…

ARCHITETTURA

Gli elementi essenziali di un'applicazione VoiceXML sono quattro (Fig.1):

- **Rete telefonica:** vale a dire il mezzo che permette all'utente di entrare in comunicazione con un server VoiceXML. Se il chiamante utilizza un telefono la rete telefonica potrebbe essere, per esempio, PSTN o GSM; nel caso di un computer dotato di microfono e collegato a Internet sarà con molta probabilità una rete VoIP (*Voice over IP*).

- **Server VoiceXML:** è il collante tra l'utente e i dati, in quanto si occupa di accettare le richieste del chiamante e di fornire "a voce" le informazioni ottenute dal server Web. In poche parole si tratta di una piattaforma su cui è in esecuzione un interprete VoiceXML.

XML

eXtensible Markup Language è un linguaggio (definito dal W3C) mediante il quale è possibile creare linguaggi di markup personalizzati. Si noti che i linguaggi di markup descrivono il formato di un documento, in altre parole il modo in cui il contenuto del documento deve essere interpretato. Esistono numerosi linguaggi basati su XML, solo per citarne alcuni: VoiceXML, SMIL, MathML, SVG, XHTML, CML, CBL, …

Web Multimedia

VoiceXML

XML fa sentire la sua voce

Fig. 1: Semplificazione dell'architettura VoiceXML: l'utente comunica a voce con il server VoiceXML. La richiesta viene inoltrata al server Web, successivamente i dati ricevuti dal server Web sono trasformati in audio dal server VoiceXML e comunicati all'utente.

- **Rete TCP/IP:** Internet o una qualsiasi rete TCP/IP collegano, quasi sempre grazie al protocollo HTTP, il server VoiceXML ad un server Web.

- **Server Web:** contiene fisicamente i dati e risponde alle richieste del server VoiceXML. Mi preme sottolineare che il server web può essere sostituito da un generico dispositivo (*Application server*) idoneo a soddisfare le richieste del server VoiceXML.

Dunque la grande differenza rispetto ad una normale sessione Web (Fig. 2) è data dalla presenza del server VoiceXML. Riassumiamo brevemente in che modo si concretizza l'interazione uomo-macchina nel modello illustrato:

1) L'utente crea una connessione con un server VoiceXML semplicemente chiamando un numero telefonico (o tramite Internet);

2) Ad ogni chiamata ricevuta, il server VoiceXML interpreta un file con estensione *.vxml* ossia un banalissimo file testuale contenente codice XML. Nel prossimo paragrafo scopriremo come scrivere codice vxml. In base ai comandi vxml interpretati il server può: porre domande o in alternativa inviare messaggi, suoni, musica all'utente, accettare comandi per mezzo del riconoscimento vocale, registrare le richieste dell'utente senza riconoscere le parole.

3) I file *.vxml* possono contenere script implementati con linguaggi esterni, a titolo di esempio Javascript ma con un po' di lavoro in più anche Delphi, C++, … In ogni caso un server VoiceXML deve interrogare un application server, normalmente un web server, al fine di ottenere i dati da restituire all'utente sotto forma di messaggi preregistrati o prodotti da sistemi di sintesi vocale.

Fig. 2: L'utente richiede una pagina Web ad un server.

DTMF

Dual Tone Multi Frequency è un metodo per comunicare con un sistema, durante una chiamata telefonica, attraverso la pressione di tasti. Ad ogni tasto sono associati simultaneamente due toni, uno per la riga ed uno per la colonna. Messaggi tipo *"Prema il tasto 1 per avere informazioni su..."* si basano sui toni DTMF.

Non temete, le cose sono molto più semplici di quello che sembrano: così come un progettista di pagine Web non deve preoccuparsi di router, gateway, protocolli di rete, un programmatore VoiceXML può tranquillamente ignorare la complessità delle tecnologie telefoniche e dei sistemi di riconoscimento/sintesi vocale. In qualità di programmatori dobbiamo conoscere solo la sintassi che consente di creare applicazioni VoiceXML.

SINTASSI

Purtroppo lo spazio a mia disposizione è limitato, di conseguenza, nella speranza di stimolare il vostro appetito, non potrò che fornirvi un assaggio delle funzionalità di VoiceXML. Un documento VoiceXML costituisce una macchina a stati finiti, l'utente in un dato momento è impegnato in una ed una sola conversazione (o dialogo). Gli elementi più importanti del linguaggio sono elencati nella Tab. 1. Analizziamo un esempio semplice ma significativo di codice VoiceXML:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<meta name="author" content="Salvatore Meschini"/>
<menu>
<prompt>Scegli il tuo linguaggio di programmazione
        preferito tra <enumerate/></prompt>
<choice next="delphi.vxml"> delphi </choice>
<choice next="cs.vxml"> c sharp  <choice>
<choice next="java.vxml"> java <choice>
<help>
Per avere accesso alle informazioni relative ad un
 particolare argomento dimmi quale è il tuo linguaggio di
 programmazione preferito tra Delphi, C# e Java. Grazie!
</help>
<noinput>Non ho sentito nulla, per favore dammi una
        risposta</noinput>
<nomatch>Per favore parla più chiaramente, la tua
        risposta non ha senso</nomatch>
</menu>
</vxml>
```

Ogni documento VoiceXML deve iniziare con l'attri-

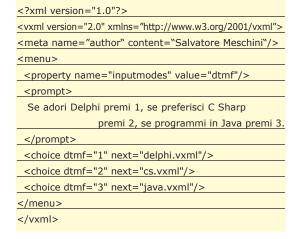
<assign>	Assegna un valore ad una variabile
<audio>	Riproduce un file audio
<block>	Contenitore di codice eseguibile non-interattivo
<catch>	Intercetta un evento
<choice>	Definisce un elemento di menu
<clear>	Elimina una o più variabili
<disconnect>	Termina una sessione
<else>	Struttura condizionale else
<elseif>	Struttura condizionale elseif
<enumerate>	Elenco di scelte di un menu
<error>	Intercetta un errore
<exit>	Esci da una sessione
<field>	Dichiara un campo per l'input dei dati
<filled>	Azione da eseguire quando i campi sono riempiti
<form>	Un attributo per ottenere e fornire informazioni
<goto>	Cambia il dialogo corrente
<grammar>	Specifica da grammatica da utilizzare
<help>	Intercetta un evento "richiesta di aiuto"
<if>	Struttura condizionale if
<initial>	Dichiara il comportamento iniziale
<link>	Specifica una transizione comune a tutti i dialoghi
<log>	Genera un messaggio di debug
<menu>	Un dialogo per la scelta tra varie alternative
<meta>	Definisce un metadato sotto forma di coppia
<metadata>	Definisce le informazioni metadata
<noinput>	Cosa succede in mancanza di input?
<nomatch>	Cosa succede se non si trovano corrispondenze?
<object>	Interagisci con un'estensione personalizzata
<option>	Specifica un'opzione in un <field>
<param>	Parametro in <object> oppure <subdialog>
<prompt>	Legge del testo o riproduce un file audio
<property>	Controlla le impostazioni della piattaforma
<record>	Registra la voce del chiamante
<reprompt>	Riproduci un <prompt> dopo un evento
<return>	Termina un sottodialogo (sottoprocedura)
<script>	Utilizza uno script lato client di tipo ECMAScript
<subdialog>	Invoca un sottodialogo
<submit>	Invia dei valori ad un server
<throw>	Genera un evento.
<transfer>	Trasferisci il chiamante ad altra destinazione
<value>	Inserisci un valore in un <prompt>
<var>	Dichiara una variabile
<vxml>	Elemento che descrive il documento VoiceXML

Tab. 1: **I Tag definiti da VoiceXML.**

buto *<vxml>* contenente almeno le informazioni sulla versione e sul namespace, cioè *xmlns="http://www.w3.org/2001/vxml"*. Altri attributi sono opzionali. L'intera conversazione è racchiusa in un blocco *<menu></menu>* e prevede la scelta fra tre opzioni definite mediante l'attributo *<choice>*. Ad ogni scelta è associata una transizione verso un diverso documento con estensione *.vxml*. In parole povere il server, in base alla parola pronunciata dall'utente, eseguirà altri comandi VoiceXML fornendo informazioni o ponendo nuove domande. Il codice proposto prevede la gestione della mancanza di input da parte del chiamante *<noinput>*, l'ingresso di dati non compresi dal sistema *<nomatch>* e la richiesta esplicita di aiuto *<help>*. La riproduzione di un file audio è avviata quando l'interprete incontra un tag *<audio>*. Chiamando il numero di telefono associato al servizio si potrebbe verificare un'interazione del genere:

Computer: *Scegli il tuo linguaggio di programmazione preferito tra Delphi, C Sharp, Java*
Utente: (silenzio)
Computer: *Non ho sentito nulla, per favore dammi una risposta*
Utente: Visual Basic
Computer: *Per favore parla più chiaramente, la tua risposta non ha senso*
Utente: Help
Computer: *Per avere accesso alle informazioni relative ad un particolare argomento dimmi quale è il tuo linguaggio di programmazione preferito tra Delphi, C# e Java. Grazie!*
Utente: Delphi

Come potete osservare con pochissime righe di codice abbiamo realizzato un portale vocale funzionante, anche se al momento assolutamente inutile. Un altro metodo per ricevere input da un utente durante una chiamata è dato dai toni DTMF, l'esempio precedente viene trasformato in:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<meta name="author" content="Salvatore Meschini"/>
<menu>
  <property name="inputmodes" value="dtmf"/>
  <prompt>
   Se adori Delphi premi 1, se preferisci C Sharp
                    premi 2, se programmi in Java premi 3.
  </prompt>
  <choice dtmf="1" next="delphi.vxml"/>
  <choice dtmf="2" next="cs.vxml"/>
  <choice dtmf="3" next="java.vxml"/>
</menu>
</vxml>
```

L'uso delle grammatiche in VoiceXML meriterebbe un articolo a parte, vi basti sapere che il loro compito consiste nel garantire una corretta interpretazione semantica delle parole e delle sequenze DTFM. Una grammatica creata in modo appropriato, per mezzo del blocco *<grammar></grammar>*, rende più naturale e flessibile il dialogo tra l'utente ed il sistema vocale. I formati per la definizione di una grammatica sono due: quello standard W3C e il GSL di Nuance. Vi rimando alle oltre duecento pagine delle specifiche ufficiali Voice Extensible Markup Language Version 2.0 per una descrizione molto dettagliata di tutti gli elementi della sintassi con tanto di esempi a corredo. Dopo aver scritto del codice VoiceXML è necessario testarlo con appositi browser, in tal caso procuratevi un buon sistema di sintesi text-to-speech, o servendovi dei servizi di hosting.

GLI STRUMENTI

Ogni linguaggio di programmazione è supportato da strumenti di sviluppo più o meno potenti, VoiceXML non fa eccezione a questa regola. E' però necessario distinguere gli IDE (Integrated Development Environment) dagli ambienti hosted. Alla prima categoria appartengono i prodotti software che integrano editor di testo, compilatore, debugger ed altri componenti utili

HOSTING

Dopo aver sviluppato un'applicazione VoiceXML probabilmente sorgerà il bisogno di trovare server telefonici, gateway e spazio per rendere pubblico il nostro servizio. Società quali BeVocal, VoiceGenie, TellMe, VoxPilot, Voxeo, HeyAnita offrono tutto il necessario. Cercando con Google i rispettivi siti si può accedere ad applicazioni VoiceXML dimostrative.

VoiceXML
XML fa sentire la sua voce

JTAPI

Il linguaggio Java mette a disposizione del programmatore il pacchetto *javax.telephony*, ricco di metodi per gestire chiamate, fornitori di servizio, numeri telefonici e terminali. La documentazione completa di JTAPI è scaricabile dal sito http://java.sun.com

Sul Web

http://www.w3.org/TR/voicexml20/
Documentazione ufficiale di VoiceXML 2.0

http://salvatoremeschini.cjb.net
Codice sorgente VoiceXML e non solo

http://fife.speech.cs.cmu.edu/openvxi/
SpeechWorks OpenVXI - l'interprete VoiceXML di riferimento (open-source)

http://gin2.itek.norut.no/elvira/_elvira.php?p=introduction
Elvira - browser vxml

http://www.nuance.com
WebServer vocale

http://www.microsoft.com/speech
Speech SDK per applicazioni Windows, sintesi text-to-speech

Fig. 3: BeVocal: uno dei tanti fornitori di servizio. Chiamando il numero (+44) 2079613985. Potete testare le vostre applicazioni (attenti alla bolletta!).

in fase di sviluppo. Se facciamo riferimento al linguaggio VoiceXML la seconda categoria è formata essenzialmente dai fornitori di servizio on-line che trovate nel box a bordo pagina. Nella scelta di uno strumento di sviluppo VoiceXML esistono diversi criteri da tenere in considerazione:

- **Meccanismi di accesso:** ovvero attraverso quale mezzo è possibile accedere all'applicazione, per fare qualche esempio rete telefonica mobile e/o fissa, Voice over IP, simulatori;
- **Grammatica supportata:** l'ambiente deve prevedere la creazione e la gestione di grammatiche possibilmente compatibili con il formato W3C XML oppure Nuance GSL. E' opportuno assicurarsi che il prodotto sia aggiornato alla versione 2.0 di VoiceXML!
- **Progettazione visuale:** per non complicarvi la vita puntate su IDE che consentono di progettare visualmente il flusso delle chiamate, magari mediante componenti riusabili;
- **Compatibilità:** se possibile orientatevi verso IDE compatibili con il servizio di hosting scelto;

Una buona opzione è rappresentata da IBM WebSphere Voice Toolkit dal momento che si integra perfettamente nella piattaforma Eclipse. Se invece siete interessati all'implementazione di un interprete VoiceXML vi suggerisco di dare un'occhiata al programma open-source OpenVXI.

SAPORE DI SALT

Un gruppo di società guidato da Microsoft e Intel ha proposto al W3C le specifiche di SALT *(Speech Application Language Tags)*, un progetto teso alla creazione di un linguaggio per applicazioni multimodali, cioè fruibili contemporaneamente da diversi sensi, e all'integrazione del parlato nella pagine Web. Se un giorno potremo comunicare a voce con un palmare nuovo fiammante forse sarà merito anche di SALT. Pur avendo finalità diverse SALT è utilizzabile anche nel mondo della telefonia e può quindi essere giudicato un concorrente di VoiceXML. Microsoft ha rilasciato di recente un kit di sviluppo chiamato *.NET Speech SDK* che potrebbe mettere i bastoni tra le ruote a VoiceXML, quanto meno su piattaforma Windows. Attualmente però SALT, definibile come un'estensione di HTML e XHTML, non raggiunge la completezza e la maturità di VoiceXML.

CONCLUSIONI

E' da tempo che vi chiedete cosa si nasconda dietro ai famigerati "servizi clienti" dei gestori di telefonia mobile? Avete sempre desiderato progettare e realizzare un portale vocale? Cercate un settore in cui specializzarvi? Spero che questo articolo abbia risposto almeno in parte alle vostre curiosità. Vi consiglio caldamente di provare le applicazioni dimostrative delle varie società di hosting elencate nel box a bordo pagina. VoiceXML è un linguaggio molto potente per creare e gestire dialoghi ma non prevede funzioni per il controllo di chiamata (routing, multiconferenza, composizione di nuove chiamate, …), a tale mancanza sopperisce il linguaggio *Call Control eXtensible Markup Language, CCXML* per gli amici. Ma questa è tutta un'altra storia… Che prima o poi vi racconteremo!

Salvatore Meschini

Voice Enabling Web Applications: VoiceXML and Beyond

Editore: Apress www.apress.com
Autore: Ken Abbott
Anno: Novembre 2001
ISBN: 1-893115-73-9
Pagine: 250
Lingua: Inglese
Contiene 1 CD-Rom
Prezzo: $34.95

VOICE ENABLING WEB APPLICATIONS: VOICEXML AND BEYOND è un ottimo volume, in lingua inglese, che mostra quelli che sono i concetti, le strategie, l'architettura di Voice XML, la nuova tecnologia, basata sullo standard XML, per la gestione vocale dei contenuti Web. Il testo mostra dei pratici tutorial che guidano l'utente, passo passo, nella progettazione e gestione di pagine Web basate su Voice XML; più in particolare viene fatto riferimento ad applicazioni web multi-tier con supporto di Apache Cocoon. L'intero volume è suddiviso in tre distinte parti, ognuna delle quali mostra delle componenti fondamentali del linguaggio; nella fattispecie, la prima parte è rivolta a chi vuoi acquisire nozioni dettagliate su VoiceXML: come funziona il riconoscimento e la sintesi vocale e come integrare tali funzionalità con un linguaggio come XML. La seconda parte illustra la sintassi e i concetti fondamentali del linguaggio, mentre la parte finale è un vero e proprio tutorial guidato su come integrare, all'interno di un sito Web, applicazioni Voice XML, analizzandone tutti gli aspetti, anche di integrazione a larga scala. Il CD-Rom allegato, contiene tutte le applicazioni necessarie per iniziare a sviluppare applicazioni.

Un oscilloscopio

‘DIGITALE’ IN DELPHI

Sfruttando le connessioni di una porta seriale è possibile realizzare un efficace oscilloscopio per livelli logici digitali. Il programma accede in profondità alle qualità hardware del Computer, andando ad esplorare le caratteristiche più intime della macchina, interfacciandosi direttamente con i componenti hardware a basso livello. Con questo software è possibile utilizzare un qualunque Personal Computer per svolgere le operazioni possibili soltanto con un oscilloscopio.

Ogni programma, qualunque esso sia ed indipendentemente dal linguaggio di programmazione utilizzato, necessita prima o poi di una azione di verifica e di debug.
Nel campo della progettazione elettronica, in seguito alla realizzazione di un prototipo, oppure in fase di ricerca dei guasti, si effettua un'opera di verifica del buon funzionamento del circuito.
Al contrario della programmazione, però quasi sempre è impossibile fermare le operazioni che avvengono all'interno di un componente elettronico per analizzarne il comportamento, come possiamo fare durante il debug di un programma, per verificarne ad esempio i valori delle varie variabili.
Se volessimo fare un paragone tra la maggior parte dei circuiti elettronici e la programmazione, potremmo dire che all'interno di una apparecchiatura elettronica si è quasi sempre in ‘runtime’, perciò l'analisi dei comportamenti dei prototipi andrà effettuata con diversi strumenti di misura, tra i quali riveste sicuramente una posizione di assoluta importanza l'oscilloscopio.
L'oscilloscopio ha la capacità di visualizzare su uno schermo l'andamento di un segnale elettrico in funzione del tempo: nel nostro caso ci proponiamo di analizzare i segnali in termini di valore logico (‘0’ oppure ‘1’ logico), caratteristica fondamentale nell'elettronica digitale.
La limitazione derivante dal fatto che si possa ‘leggere’ soltanto il valore logico di una data linea, viene compensata dal fatto che con questo programma siamo in grado di visualizzare quattro ingressi e che abbiamo la possibilità di comandare due uscite, mentre con un oscilloscopio classico generalmente si hanno a disposizione soltanto due ingressi.
In queste pagine vedremo come sia possibile, senza alcuna aggiunta di hardware esterno, realizzare un analizzatore logico a quattro canali di ingresso e due in uscita, che ci sarà utile per l'analisi e la realizzazione di svariate apparecchiature che verranno proposte in futuro.

LA PORTA SERIALE

A dispetto del moltiplicarsi di svariati sistemi di controllo e comunicazione, sempre più veloci e potenti, esistono alcune porte per Personal Computer che sono rimaste praticamente invariate per decenni.
Stiamo parlando ovviamente della porta seriale, che insieme alla parallela, hanno mantenuto immutato il loro aspetto esterno e che hanno variato non di molto, nella sostanza della loro funzionalità quello interno, al di là ovviamente della maggiore miniaturizzazione dei componenti.
In questa fase vorrei dare al lettore una idea di massima della struttura interna di una porta seriale, senza scendere in troppi dettagli tecnici, viste le limitazioni di spazio a disposizione per la trattazione.
Tralasciando quindi le ovvie differenze con la porta parallela, ci limitiamo a dire che la porta di comunicazione seriale ha lo scopo di scambiare informazioni con un'altra apparecchiatura, mediante un apposito protocollo di trasmissione e per mezzo di connessioni opportune: lo scambio dei dati è in ogni caso seriale, ovverosia un bit per volta.
La conversione parallelo-seriale prima della trasmissione e seriale-parallelo dopo la ricezione avviene per mezzo di un particolare circuito integrato chiamato UART *(Universal Asynchro-*

Elettronica e Delphi

File sul CD
\soft\codice \elettronica.zip

Il programma completo, compilato e funzionante è disponibile nel CD allegato con il nome: *‘SpuntoLogicTester.zip’*

L'Oscilloscopio
L'oscilloscopio è uno strumento che permette di visualizzare l'andamento della tensione elettrica su uno o più canali in funzione del tempo.

Elettronica e Delphi

Oscilloscopio
'Digitale' in Delphi

nous Receiver / Transmitter). All'interno del PC, per la gestione della porta parallela esistono svariati componenti, che hanno lo scopo di controllare le varie linee di comunicazione della porta. Nella nostra applicazione, eviteremo accuratamente di interfacciarci con la UART, utilizzando i registri che ci permettono di accedere direttamente alle linee di comunicazione connesse fisicamente al connettore della nostra porta seriale. Qui di seguito vengono riportate le connessioni per la porta seriale.

Connet. 25 PIN	Connet. 9 PIN	Segnale	Tipo di segnale
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

Il connettore può essere di due tipi, DB 9 oppure DB 25, che significa semplicemente a 9 oppure 25 piedini, anche se quest'ultimo tipo di connessione sta andando in disuso a favore del DB9. Una porta seriale DB25 si riconosce facilmente da una parallela DB25 perché quest'ultima è dotata di un connettore femmina, al contrario della prima che ha un connettore maschio, guardando ovviamente il retro del PC.

La seriale come porta di Input/Output

In queste pagine vedremo come sia possibile utilizzare la porta seriale come porta di I/O per la realizzazione di un oscilloscopio per livelli logici.

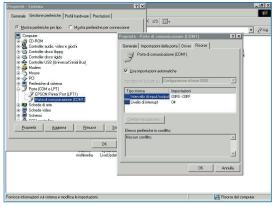



Fig. 1: Per verificare quale indirizzo base abbia la porta seriale installata sul PC è sufficiente controllare la pagina relativa alle porte COM e LPT selezionando *Pannello di Controllo/Sistema /Gestione Periferiche/PorteCOM e LPT/ COM1-2/ Risorse*, come mostrato in figura.

Occorre fare attenzione al fatto che sulle linee della porta è presente una tensione compresa tra +3 e +25 Volts, quando la linea è a livello logico '1' , mentre quando si ha un livello logico '0' sul contatto si ha un valore di tensione compreso tra -3 e –25 Volts, al contrario della maggor parte delle componenti interne del PC che sono alimentate con +5 Volts. Da quanto appena detto appare ovvio che non è possibile interfacciare direttamente una porta seriale con qualsiasi altra apparecchiatura digitale alimentata con +5V, senza utilizzare appositi circuiti adattatori (MAX 232 ad esempio). Ogni porta seriale viene individuata a livello fisico da un indirizzo, che di fatto è divenuto standard, ma che può essere facilmente controllato selezionando: *Pannello di Controllo/Sistema/Gestione Periferiche/PorteCOM e LPT/ COM1-2/ Risorse:* facciamo attenzione se abbiamo un mouse che utilizza una porta seriale, perché l'accesso diretto ai relativi canali hardware potrebbe pregiudicarne il funzionamento.

Porta	Indirizzo base	Interrupt
COM 1	3F8	4
COM 2	2F8	3
COM 3	3E8	4
COM 4	2E8	3

Solitamente gli indirizzi ed i canali di Interrupt relativi a ciascuna porta sono quelli riportati nella tabella, anche se conviene controllare la correttezza di questa affermazione attraverso la *Gestione Periferiche.*

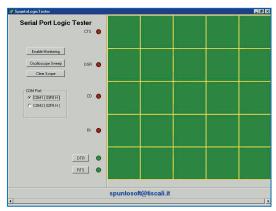



Fig. 2: Al momento dell'esecuzione del programma si accede alla schermata iniziale di figura, in questa fase il monitoring della porta è ancora disabilitato per motivi di sicurezza.

GESTIONE SOFTWARE DELLA PORTA SERIALE

La gestione software della porta seriale avviene attraverso otto registri di Input/Output ed un sistema di controllo degli interrupt, la trattazione del quale esula dallo scopo di queste pagine.Degli otto registri appena citati, per la nostra applicazione abbiamo bisogno soltanto

Indirizzo	Accesso	Registro	Significato
Base + 3	Read/Write	LCR	Line Control Register
Base + 4	Read/Write	MCR	Modem Control Register
Base + 6	Read	MSR	Modem Status Register

dei tre che vengono riportati in tabella e dei quali diamo una descrizione sommaria (Tab. 3) Il registro LCR *(Line Control Register)* ha il compito di controllare i parametri di comunicazione base della porta, in particolare il bit 6, quando è attivo forza TD in una condizione di 'spacing', che equivale ad un livello logico '1'. *Modem Control Register* (MCR) è un registro di lettura/scrittura ed ha lo scopo di controllare un eventuale modem connesso alla porta; a questo scopo utilizziamo il bit 0 di questo registro, che ci permette di gestire la linea DTR *(Data Terminal Ready)*, nonché il bit '1' relativo a RTS *(Request to Send)*. Infine il registro di sola lettura MSR *(Modem Status Register)*, relativo allo status del Modem, ci permette di accedere alle linee individuate dai bit 4-7 corrispondenti a CTS, DSR, RI e CD che saranno il fulcro della nostra applicazione.

IL PROGRAMMA

Dopo la breve descrizione delle caratteristiche hardware e software della porta, siamo giunti finalmente all'analisi del codice del programma.

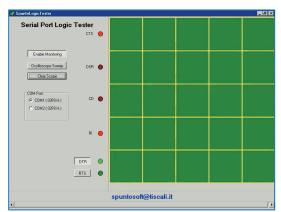



Fig. 3: **Premendo il pulsante 'Enable Monitoring', si accede ai valori logici della porta, con un 'update rate' di 10 millisecondi: lo stato logico delle linee di ingresso viene visualizzato per mezzo di LED rossi, mentre per quelle di uscita con LED verdi.**

Nell'intestazione notiamo che vengono utilizzati componenti standard di Delphi, reperibili anche nella versione Personal, fatta eccezione per *'SpuntoLedComponent'* che è comunque disponibile nel CD allegato sotto il nome di *'SpuntoLedComponent.dcu'* e che può essere tranquillamente installato come un qualunque altro componente Delphi.

```
unit SpuntoLogicTesterUnit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
                        Graphics, Controls, Forms,
  Dialogs, ExtCtrls, SpuntoLedComponent, jpeg,
                        Buttons, StdCtrls;
```

Il tipo *'TPortArray'* ha lo scopo di contenere la decodifica binaria del valore decimale che viene letto da un determinato registro e di contenere la codifica da inviare alla porta in uscita.

```
type
//********** Port related arrays **************//
  TPortArray= Array[0..7] of Boolean;
                              // Array of Port bits
```

La classe principale, che contiene tutte le funzionalità del programma, ingloba un componente TTIMER, che ha la funzione di scandire l'avanzamento della spazzata dell'oscilloscopio e di garantire l'aggiornamento dei LED visualizzati sulla finestra del programma. Le procedure e le funzioni della classe vengono descritte brevemente più avanti, mentre per quanto riguarda le variabili globali di programma, possiamo citare *CombaseAddress*, contenente l'indirizzo base della porta seriale, ed i relativi valori dei tre registri che utilizziamo contenuti in *MCRAddress*, *LCRAddress* e *MSRAddress*; infine *RTS*, *CTS*, *DSR*, *CD* e *DTR* contengono lo stato logico della corrispondente linea fisica della porta seriale. Nel CD: *\soft\codice\elettronica\mainclass.txt*. Alla creazione della finestra viene eseguita la procedura *FormCreate*, che inizializza le variabili globali di programma, definendo la porta in uso per default (nel nostro caso *COM1*), i relativi indirizzi dei registri in uso, nonché la predisposizione delle variabili per il controllo grafico della spazzata dell'oscilloscopio. Esaminiamo la form creata presente in*\soft\codice\elettronica\formcreate.txt*. Alla pressione del tasto *'Enable Monitoring'* viene eseguita la procedura che segue, che si occupa di abilitare o disabilitare il timer, avviene la lettura di tutte le porte e ven-

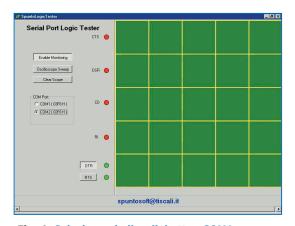



Fig. 4: **Selezionando il radiobutton COM1, oppure COM2 è possibile analizzare l'una o l'altra porta: occorre fare attenzione, come è già stato detto a selezionare la porta installata sul proprio PC.**

Elettronica e Delphi

Oscilloscopio
'Digitale' in Delphi

Livelli logici sulla porta seriale

Sui contatti fisici del connettore della porta seriale è presente una tensione compresa tra +3 e +25 Volts quando la linea è a livello logico '1', mentre quando si ha un livello logico '0' sulle linee si ha un valore di tensione compreso tra -3 e –25 Volts, quindi non direttamente interfacciabile con altri circuiti logici alimentati con +5 Volts.

Elettronica e Delphi

Oscilloscopio
'Digitale' in Delphi

gono posizionati i tasti corrispondenti alle linee di uscita DTR ed RTS.

```
procedure TSpuntoLogicTester.EnableMonitoringClick
                                        (Sender: TObject);
begin
Timer1.Enabled:=EnableMonitoring.Down;
 ReadAllPorts;
 If DTR then DTRSpeedButton.Down:=True else
                                 DTRSpeedButton.Down:=False;
 If RTS then RTSSpeedButton.Down:=True else
                                 RTSSpeedButton.Down:=False;
end;
```

Le procedure *WritePort* e *ReadPort* sono presenti sul CD: *\soft\codice\porteIO.txt*, scritte in parte in linguaggio Assembler, si occupano di leggere e scrivere i registri di Input/Output della porta seriale, accedendo direttamente al componente fisico, per questo motivo, potrebbero verificarsi errori di *'Privileged error'*, utilizzando WIN XP oppure WIN 2000, i quali operano un controllo diretto sulle operazioni di I/O. Le procedure *RadioButtonCOM1Click* e *RadioButtonCOM2Click*, operano la ridefinizione degli indirizzi dei registri da utilizzare, alla pressione del relativo radiobutton. (*\soft\codice\elettronica\radiobutton.txt*). La lettura di tutti i registri interessati alla gestione del nostro oscilloscopio logico avviene con la procedura *ReadAllPorts*, leggendo prima il valore delle porte fisiche attraverso la procedura *ReadPort* e poi codificandone il valore nel relativo Array binario per poi estrarre il singolo valore logico della linea fisica collegata alla porta.

```
procedure TSpuntoLogicTester.ReadAllPorts;
//Readr all COM Ports related to selected serial port
Var
  MCRWord,LCRWord,MSRWord:Word;
Begin
  MCRWord:=Readport(MCRAddress);
  LCRWord:=Readport(LCRAddress);
  MSRWord:=Readport(MSRAddress);
  ExtractPortArray(MCRWord,MCR);
  ExtractPortArray(MSRWord,MSR);
  ExtractPortArray(LCRWord,LCR);
  RTS:=MCR[1];        // OUT
  CTS:=MSR[4];        // IN
  DSR:=MSR[5];        // IN
  CD :=MSR[7];        // IN
  DTR:=MCR[0];        // OUT
  RI :=MSR[6];        // IN
End;
```

Il componente *Timer*, quando è attivo, ha il compito di eseguire periodicamente la procedura che segue, nella quale vengono aggiornati gli stati dei LED, corrispondenti ai valori logici delle singole linee e se viene richiesta anche la visualizzazione grafica sullo schermo dell'oscilloscopio, viene operato lo 'sweeping' grafico in modo parallelo, per le quattro linee in ingresso e le due in uscita. Esaminiamo la procedura Timer1Timer (*\soft \codice\elettronica\Timer1Timer.txt*). Le procedure *ExtractPortArray* e *EncodePortarray* si occupano rispettivamente della decodifica e della codifica del valore decimale letto dal registro, nell'array *PortArray*, sotto forma di un vettore di valori Booleani, che facilita notevolmente la gestione dei pulsanti corrispondenti alle linee di uscita e dei LED per quelle di ingresso.

```
Procedure TSpuntoLogicTester.ExtractPortArray(
                  PortByte:Word; Var PortArray:TPortArray);
Var
I,Mask:Word;
begin
//Decodes the 'PortByte' word into the 'PortArray' Array
Mask:=1;
  For I:=0 to 7 do begin
    Portarray[I]:=(PortByte and Mask)>0;
                                    // Extracts the data bits
    Mask:=Mask shl 1;
  End;
end;
Procedure TSpuntoLogicTester.EncodePortArray(
                  Var PortByte:Word; PortArray:TPortArray);
Var
I,Mask:Word;
begin
//Encodes the 'PortArray' Array into the 'PortByte' word
PortByte:=0;
Mask:=1;
  For I:=0 to 7 do begin
    if PortArray[I] then PortByte:=PortByte+Mask;
    Mask:=Mask shl 1;
  End;
end;
```

La pressione dei tasti *DTR* e *RTS*, provoca l'e-

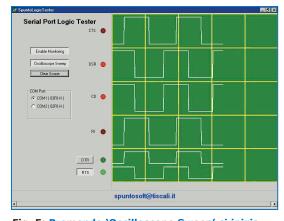



Fig. 5: Premendo 'Oscilloscope Sweep' si inizia l'acquisizione grafica degli stati logici delle linee di I/O della porta.

Sul WEB
Ulteriori informazioni sono reperibili sul sito:

http://web.tiscali.it/spuntosoft/

secuzione degli event handler che seguono: viene quindi letta la posizione del tasto, correlata al relativo bit del registro appropriato, codificato sotto forma di una Word, per renderne possibile l'invio alla porta tramite la procedura *WritePort:*

```
procedure TSpuntoLogicTester.DTRSpeedButtonClick(
                                   Sender: TObject);
VAR
MCRWord:Word;

begin
ReadAllPorts;
MCR[0]:=DTRSpeedButton.Down;
//Codifico MCR in Word e poi lo invio alla porta
EncodePortArray(MCRWord,MCR);
WritePort(MCRAddress,MCRWord);
end;
procedure TSpuntoLogicTester.RTSSpeedButtonClick(
                                   Sender: TObject);
VAR
MCRWord:Word;

begin
ReadAllPorts;
MCR[1]:=RTSSpeedButton.Down;
//Codifico MCR in Word e poi lo invio alla porta
EncodePortArray(MCRWord,MCR);
WritePort(MCRAddress,MCRWord);
end;
```

Infine la procedura ClearScopeClick provoca la cancellazione della griglia dello schermo dell'oscilloscopio, caricando il relativo file bitmap contenente uno schermo verde dotato di una griglia che facilita la lettura dello schermo.

```
procedure TSpuntoLogicTester.ClearScopeClick(
                                   Sender: TObject);
begin
Image1.Picture.LoadFromFile('500x500verde.bmp');
end;
```

INSTALLAZIONE ED UTILIZZO DEL PROGRAMMA

Il programma è funzionante, completo in ogni parte, corredato di codice sorgente e reperibile sul CD allegato alla rivista nel file *'SpuntoLogicTester.zip'*. Per l'installazione è sufficiente estrarre tutti i files in un'unica directory e lanciare il programma eseguibile *SpuntoLogicTesterProject.exe:* l'utilizzo è abbastanza intuitivo e viene illustrato nelle varie immagini comprese in questo articolo: occorre fare attenzione a non accedere ad indirizzi fisici dei quali non si sappia con esattezza il significato, dal momento che scrivendo a basso livello su indirizzi di I/O particolari, si possono creare effetti imprevedibili, anche il blocco del sistema. Per l'utilizzo del codice sorgente, occorre installare prima il file *'SpuntoLedComponent.dcu'* per poi aprire il progetto *'SpuntoLogicTesterProject.dpr*, l'autore sarà grato ai lettori che vogliano inviare commenti o migliorie al programma, all' indirizzo *spuntosoft@tiscali.it.*

Fig. 6: **Al termine dell'acquisizione, si avrà sullo schermo l'andamento degli stati logici delle linee della porta seriale in funzione del tempo, anche per segnali non periodici, come appare evidente in figura.**

CONCLUSIONI

Abbiamo visto brevemente, per motivi di spazio, quali sono le caratteristiche della porta seriale, sia dal punto di vista hardware che software.

È stato analizzato inoltre un programma che permette di leggere e visualizzare sullo schermo del PC lo stato logico di quattro canali di ingresso in modo istantaneo, per mezzo di LED e sotto forma di rappresentazione grafica, analogamente a quanto avviene in un oscilloscopio. Il programma è ovviamente molto semplice nella sua struttura ed ha bisogno di ampliamenti e miglioramenti, come ad esempio un controllo della velocità di campionamento: è tuttavia di immediato utilizzo ed affidabile nel funzionamento. Nonostante sia stato fatto ogni sforzo possibile teso alla verifica di quanto esposto in questa sede, il lettore comprenderà che l'utilizzo delle tecniche, del codice e delle informazioni di carattere elettronico discusse in quest'articolo, deve essere effettuato con estrema cautela, dal momento che l'autore e l'editore non possono essere accreditati di alcuna responsabilità implicita od esplicita conseguente dall'uso e dal funzionamento del software e dell'hardware esposto in questa sede.

Luca Spuntoni
(spuntosoft@edmaster.it)

Elettronica e Delphi

Oscilloscopio
'Digitale' in Delphi

Sistema Operativo

Il programma richiede un sistema operativo WIN 9X oppure Millennium.

LE FAQ DI IOPROGRAMMO

Le risposte alle domande più frequenti

Ogni mese troverete riportate le domande che più spesso giungono in redazione. Capita frequentemente che, affrontando linguaggi in continua evoluzione, si diano per scontati alcuni concetti o alcune caratteristiche di base: queste pagine sono l'occasione per ribadire o spiegare meglio tali nozioni.

.NET

Cosa serve per programmare in .NET?

Per iniziare a programmare in .NET è necessario installare il Framework .NET SDK (disponibile gratuitamente presso il sito Microsoft) che consente lo sviluppo nei vari linguaggi supportati. Tra le varie risorse include: dei compilatori, una serie di strumenti utili per effettuare test e debugging, tool per la distribuzione, configurazione, protezione e sicurezza, e altre utilità di carattere generale.
Un qualsiasi editor permette la scrittura del codice sorgente, per cui il framework SDK è sufficiente per programmare; per la realizzazione delle interfacce grafiche tuttavia, può essere di grande aiuto l'impiego di un ambiente di sviluppo RAD (Rapid Application Development) come Visual Studio .NET che già include il suddetto Framework. Se si intende programmare per il web mediante ASP .NET, dovrà essere presente sul computer anche IIS (Internet Information Server) o analogamente il Web Server Cassini interno all'IDE ASP .NET Web Matrix (anch'esso gratuito).
Per quanto concerne l'accesso ai dati e l'approccio ADO.NET, è raccomandata l'installazione dei Microsoft Data Access Components 2.7.

Quali ambienti di sviluppo esistono per .NET?

Negli ultimi mesi molte Software House si sono interessate alla realizzazione di ambienti di sviluppo per Microsoft .NET, cercando di creare delle valide alternative a Visual Studio.
Allo stato attuale, prodotti commerciali di terze parti non giustificano la spesa economica, poiché le alternative migliori restano comunque quelle derivanti da progetti open source per la creazione di ambienti di sviluppo gratuiti. Gli IDE maggiormente utilizzati sono:

- *Microsoft Visual Studio .NET:* indubbiamente il migliore strumento per lo sviluppo sia di applicazioni Windows, sia per il Web. Permette l'uso di più linguaggi ed è disponibile in tre versioni: Professional, Enterprise Developer ed Enterprise Architect. Il prezzo non è propriamente contenuto, ma se ci si limita alla programmazione con un solo linguaggio è possibile scegliere una versione ridotta dell'ambiente: Visual C# Standard, Visual Basic .NET o ancora Visual C++ .NET Standard. Il costo di quest'ultimi è più abbordabile e non arriva ai 150 Euro. Una nuova versione di Visual Studio sta per fare la sua entrata: Visual Studio .NET 2003.

- *Sharp Develop:* ambiente di sviluppo GRATUITO e open source (licenza GPL), è un valido strumento per i linguaggi C# e Visual Basic .NET. Attualmente il supporto per la realizzazione delle interfacce grafiche (windows forms) non è paragonabile a quello di Visual Studio .NET, ma è un ambiente che viene costantemente migliorato ed è molto diffuso.

- *ASP.NET Web Matrix:* ambiente di sviluppo per il Web, rilasciato gratuitamente da Microsoft; è visuale, molto ben fatto e ha al suo interno un Web Server installabile su Windows XP Home, sul quale IIS 5 non può essere installato (in realtà è possibile attraverso degli accorgimenti molto particolari).

Quali linguaggi posso usare con .NET?

Microsoft fornisce i compilatori per C#, Visual Basic .Net, Visual C++ .Net, J# e Jscript .NET. La lista dei linguaggi .NET enabled è di gran lunga maggiore (alcune decine) e in costante aggiornamento, poiché terze parti lavorano nella realizzazione di compilatori per i più comuni linguaggi, tra cui: COBOL, Eiffel, Small Talk, Perl e Python. Borland, storica concorrente di Microsoft, ha mostrato il suo interesse annunciando l'uscita per la metà del 2003 di un successore di Delphi (nome in codice Galileo), che renderà Delphi completamente compatibile con .NET e che sarà un'alternativa a Visual Studio consentendo lo sviluppo anche con gli altri linguaggi come C# o Visual Basic .NET.

Su quali sistemi operativi girano le applicazioni .NET?

Le applicazioni .NET necessitano del Framework .NET Redistributable per essere eseguite e gestite, per cui dovrà essere installato nei client per eseguire programmi .NET. Tale versione del framework ha dimensioni molto ridotte rispetto all'SDK, perché contiene solo la parte indispensabile per l'esecuzione delle applicazioni. I sistemi operativi che lo supportano sono:

- Microsoft Windows 98
- Microsoft Windows NT 4.0 (Service Pack 6a richiesta)
- Microsoft Windows Millennium Edition
- Microsoft Windows 2000 (consigliata Service Pack 2)
- Microsoft Windows XP Professional

- Microsoft Windows XP Home

Esiste una versione detta .NET Compact Framework installabile su Windows CE. Sono inoltre in corso dei porting su Linux, primo fra tutti MONO (*www.go-mono.org*).

Su quali sistemi operativi è possibile installare il Framework .NET SDK?

A differenza della versione Redistributable, l'SDK del framework .NET ha requisiti software piuttosto restrittivi, in particolare l'installazione è nominalmente possibile su:

- Microsoft Windows NT 4.0 (Richiesta Service Pack 6a)
- Microsoft Windows 2000 (Consigliata Service Pack 2)
- Microsoft Windows XP Professional

Salvo trucchi particolari, Microsoft non consente di installare l'SDK su sistemi di tipo "home", impedendo quindi lo sviluppo con quest'ultimi. In Windows NT 4.0 invece, la mancanza di IIS 5 non permette lo sviluppo ASP .NET.

Cos'è il Microsoft Intermediate Language?

MSIL (o IL) è un insieme di istruzioni (indipendenti dalla CPU) che permettono diverse operazioni tra cui il caricamento, lo scaricamento, l'inizializzazione e l'invocazione di metodi ed oggetti. Il Framework .NET compila i programmi e le librerie in codice IL. Durante la fase di esecuzione tale codice è compilato Just In Time in codice macchina, senza provocare rallentamenti eccessivi derivanti dall'interpretazione.

Cos'è il Common Language Runtime?

Il Common Language Runtime (CLR) è il "motore" del Framework e si occupa di gestire completamente le applicazioni scritte in codice .NET, (o managed code) dalla fase di caricamento fino alla fine dell'esecuzione. Quando il compilatore prende in input un codice sorgente e genera un "eseguibile", in realtà non ha generato codice nativo per quel sistema operativo e microprocessore. Le istruzioni che si ottengono sono in IL, uno tra i compiti del CLR è proprio quello di trasformare in maniera ottimale tale codice in codice nativo durante la fase di run-time dell'applicazione, attraverso la compilazione just in time. Il CLR in realtà è una Virtual Machine molto evoluta che riesce a gestire al meglio aspetti come la sicurezza, le performance, la gestione delle eccezioni, il controllo rigoroso della tipizzazione, garbage collection e molti altri aspetti "object oriented". In queste condizioni, il punto di forza non è più un linguaggio in sé, ma è l'accoppiata IL – CLR comune a tutti i linguaggi .NET. La grande interoperabilità derivante rende quasi irrilevante la scelta del linguaggio usato per sviluppare (delle differenze in realtà esistono poiché non tutti i linguaggi sfruttano pienamente le caratteristiche del CLR come invece fa, ad esempio, C#).

I programmi .NET sono decompilabili?

Un assembly è un eseguibile (o una libreria) generato dal .NET Framework, il cui codice IL è visualizzabile mediante l'utility ILDASM. Le istruzioni IL (che sono ad un livello di astrazione maggiore rispetto al codice nativo X86) e la presenza dei Metadati, sono dei punti di forza di .NET, ma semplificano anche la possibilità di Reverse Engineering. La presenza di informazioni per il debug e di nomi dei metodi e oggetti negli assemblies, rende il compito di "decompilazione" ancora più semplice. Per ovviare a questo tipo di inconveniente esistono degli obfuscator che confondono il codice, modificando in particolar modo i valori dei metadati. Un obfuscator potrebbe per esempio modificare la stringa "CheckPassword" in un'altra del tutto casuale e priva di significato per l'uomo e il decompilatore. In questo modo non si riesce ad eliminare il problema della decompilazione, ma si cerca di scoraggiare il Reverse Engineering rendendo maggiori i costi e i tempi necessari per ottenere un codice sorgente vagamente riutilizzabile. Obfuscator comuni ed interessanti, sono Aspose Obfuscator, Dotfuscator e LSW Obfuscator.

Quali applicazioni posso sviluppare con .NET?

Differentemente da quanto pensano alcuni, .NET non è esclusivamente per il Web! Il .NET Framework SDK permette infatti lo sviluppo di applicazioni Windows, Web, e per dispositivi mobili.

Cos'è il Common Type System (CTS)?

Il Common Type System è un sistema attraverso il quale .NET definisce i tipi di dati primitivi organizzati gerarchicamente. I tipi identificati diversamente dai vari linguaggi .NET possono essere considerati degli alias per i tipi definiti dal CTS. Il CTS è indispensabile per garantire l'interoperabilità tra i linguaggi.

Cos'è il Common Language Specification (CLS)?

Il Common Language Specification è un insieme di regole e costrutti ai quali i programmatori dovrebbero attenersi per creare applicazioni interoperabili dal punto di vista dei linguaggi, librerie agevolmente modificabili da altri sviluppatori e compilatori per nuovi linguaggi. Il CLS è un sottoinsieme del CTS.

Cos'è il Garbage Collector?

La Garbage Collection è un meccanismo (gestito dal Common Language Runtime), che permette di rimuovere le risorse di memoria inutilizzate dall'applicazione. Quando un oggetto non è più accessibile nel programma ed è quindi inutilizzabile, il Garbage Collector si incarica di liberare lo spazio di memoria allocato inutilmente. In .NET il GC si occupa anche della compattazione della memoria permettendo una gestione ottimale della stessa senza l'intervento diretto da parte del programmatore.

Perché passare da Visual C++ a Visual C++ .NET

In questa faq abbiamo riassunto i dieci principali motivi che possono spingere uno sviluppatore ad abbracciare il Visual C++ .NET, abbandonando il Visual C++. Il nuovo linguaggio, oltre a fornire un migliore controllo e a garantire migliori performance, aiuta i programmatori ad affrontare le nuove e più attuali esigenze. Nuove opzioni per l'ottimizzazione della compilazio-

ne, un miglioramento della conformità agli standard ANSI/ISO, grandi passi avanti nelle librerie ed il pieno supporto per la piattaforma .NET per i Web Services, per non dire del nuovissimo IDE disponibile con Visual C++ .NET. Vediamole in dettaglio:

1. **Nuove ottimizzazioni per la compilazione.** Il compilatore di Visual C++ .NET offre agli sviluppatori numerose nuove e migliorate caratteristiche nella regolazione della generazione del codice, una su tutte la cosiddetta "Whole Program Optimization", attraverso cui il compilatore recupera le informazioni da tutti i moduli coinvolti in un programma e non da un singolo modulo per volta. Una volta in possesso di queste informazioni, il compilatore può operare al meglio a beneficio del comportamento complessivo dell'applicazione.

2. **Miglioramenti nella conformità alle direttive ANSI/ISO e STL.** Il compilatore disponibile con Visual C++ .NET è quello che più si conforma al C++ standard, grazie anche al supporto per caratteristiche standard come la possibilità di ritornare tipi covarianti. Le avanzate caratteristiche disponibili con questo standard permettono agli sviluppatori di implementare i più aggiornati pattern Object Oriented, semplificando la produzione e la riusabilità del codice. Anche al libreria STL è stata migliorata con una migliore aderenza agli standard, nuove classi container ed un miglior supporto per la programmazione thread-safe.

3. **Perfetta integrazione in Microsoft .NET.** Tra tutti i linguaggi disponibili per il framework .NET, Visual C++ offre agli sviluppatori le più potenti funzionalità per tutte le tecnologie correlate a .NET. Il codice C++ esistente può facilmente essere ricompilato per .NET senza alcuna modifica, così come gli sviluppatori possono cominciare a godere della potenza di .NET attraverso la familiare sintassi del C++. I componenti già sviluppati in C++ (e anche nuovi componenti) possono facilmente essere esposti in .NET, rendendo possibile una ricca serie di soluzioni cross-language.

4. **Un nuovo e più moderno IDE.** Il nuovo IDE messo a disposizione da Microsoft presenta numerose nuove caratteristiche che aumentano la produttività degli sviluppatori e gli consentono di concentrarsi verso le nuove e più avanzate sfide della programmazione. Pienamente integrato con tutti gli altri linguaggi della famiglia .NET, l'IDE di Visual Studio .NET fornisce agli sviluppatori C++ le migliori soluzioni per quanto riguarda editor, debugger integrato, gestione delle macro, add-in e altro ancora. Tra le funzioni più avanzate si segnalano la tecnologia IntelliSense, un nuovo help dinamico, numerose finestre scorrevoli a scomparsa che semplificano e rendono più piacevole la programmazione.

5. **Runtime Check per l'identificazione dei più comuni errori di programmazione.** La nuova tecnologia Runtime Check compiler (RTC) riesce a intercettare i più comuni errori che si presentano a run time: inconsistenza dei puntatori negli stack, overrun di array locali, stack corrotti, connessioni a variabili che risultano localmente non inizializzate, perdita di informazioni dovuta ad assegnamenti a variabili "più corte". La piena integrazione con la C-Runtime Library (CRT) e con il debugger di Visual Studio .NET consente agli sviluppatori di personalizzare il meccanismo di controllo a run-time per avere il pieno controllo ed una ampia flessibilità di utilizzo.

6. **Una migliore piattaforma per il debug.** Con Visual Studio .NET, le operazioni di debug hanno fatto un sostanziale passo avanti, sia per quanto riguarda la semplicità che la completezza delle informazioni disponibili. Tra le nuove funzioni si annoverano i run-time check, le analisi "mini-dump", la possibilità di effettuare debug remoti e multiprocesso, una nuova funzionalità di "edit&continua" e moltissimo altro ancora. Da segnalare che molte delle nuove funzionalità del debugger di Visual Studio .NET funzionano anche con applicazioni generate utilizzando Visual C++ 6.0.

7. **Programmazione dichiarativa con gli attributi.** L'utilizzo degli attributi consente agli sviluppatori di produrre codice più robusto ed in modo più rapido attraverso la Interface Definition Language. Gli attributi possono essere usati nella scrittura di qualsiasi tipo di applicazione, compresi controlli ActiveX, Applicazioni Web e Web Services.

8. **Codice più robusto e sicuro.** Sempre più spesso la principale richiesta posta agli sviluppatori concerne la sicurezza e l'integrità delle applicazioni che scrivono. Benché la sicurezza non sia mai garantita al 100% la nuova opzione "/GS" del compilatore fornisce una prima barriera contro i più comuni attacchi.

9. **Migliori prestazioni per Web Services e applicazioni Web.** Grazie alle nuove classi ATL Server, presenti nella libreria ATL, gli sviluppatori hanno a disposizione delle classi che garantiscono alte prestazione per realizzare Web Services basati su ISAPI e applicazioni per il Web dinamico.

10. **MFC 7.0 e ATL 7.0.** Le nuove Microsoft Foundation Classes (MFC) 7.0 e le Active Template Library (ATL) 7.0, vanno in contro alle nuove esigenze degli sviluppatori con una lunga serie di miglioramenti che riguardano le prestazioni, la robustezza e l'usabilità del codice. Da segnalare che le due librerie possono ora essere utilizzate tranquillamente nello stesso progetto, senza che sorgano problemi di sorta.

Java

Come posso convertire una stringa in un numero?

Il metodo da usare per convertire una stringa, dipende dal formato del nu-

mero che vogliamo ottenere, Integer, Float, Double e Long. Nel codice che riportiamo si seguito è evidenziata l'estrema facilità consentita da Java per questa conversione.

```
class ConvertTest {
  public static void main (String args[]) {
    String str;
    str = "25";
    int i = Integer.valueOf(str).intValue();
    System.out.println(i);
    long l = Long.valueOf(str).longValue();
    System.out.println(l);
    str = "25.6";
    float f = Float.valueOf(str).floatValue();
    System.out.println(f);
    double d = Double.valueOf(str).doubleValue();
    System.out.println(d);
  }
}
```

Confrontare due stringhe

Per confrontare due stringhe in Java non cadete nella tentazione di scrivere, per esempio:

```
String s1 = "Federico";
String s2 = "Federico";
if ( s1 == s2 ) { // Questo non funziona! }
```

in quanto l'operatore di confronto == verifica il riferimento delle variabili allo stesso oggetto (un po' come confrontare due puntatori in C++) e non l'uguaglianza dei valori dei due oggetti. Per un confronto valido tra due stringhe, la condizione da testare è

```
s1.equals(s2)
```

Comunicazione tra applet

Due applet in una stessa pagina HTML possono scambiarsi informazioni avendo la possibilità di accedere una all'istanza dell'altra con il metodo getApplet() dell'AppletContext, cui si deve passare il nome dell'applet come specificato dall'attributo NAME del tag APPLET.

```
Applet appUno = getAppletContext(
                        ).getApplet("AppletUno");
```

Memoria di sistema

Nell'eseguire una macchina virtuale Java (java.exe) è possibile impostare i parametri di memoria minima e massima che il PC potrà dedicare alla JVM tramite i due parametri –Xms e –Xmx, che rappresentano rispettivamente la dimensione di heap iniziale e quella massima raggiungibile, in questo modo

```
java –Xms 64 –Xmx 128 MyMainClass.class
```

I valori devono chiaramente essere compatibili con la memoria fisica presente sul sistema.

Espressioni regolari

Potete utilizzare le espressioni regolari (simili a quelle del Perl) anche in Java a partire dalla versione JDK 1.4 utilizzando il package java.util.regex, mentre per le versioni precedenti potrebbe esservi utile una libreria gratuita scaricabile dal sito *www.javaregex.com*.

Inizializzatore statico

È possibile inizializzare una classe relativamente ai suoi attributi statici utilizzando un costruttore statico, invocato una volta sola per ogni classe e prima che questa venga istanziata. La sintassi è la seguente:

```
class StaticInitClass {
   private static int x;
   private static String s1;
   static {
              x = 0;
              s1 = "Federico Mestrone";
           }
}
```

Questo consente, volendo, di inizializzare i membri statici secondo algoritmi più complessi che una semplice assegnazione.

Suddividere una stringa

Per suddividere una stringa in base ad un carattere separatore si può usare l'oggetto *java.util.StringTokenizer*, in questo modo

```
StringTokenizer tkns = new StringTokenizer(
                "Federico/Mestrone/29/TO","/");
while ( tkns.hasMoreTokens ) {
    String s = tkns.nextToken();
    // Fai qualcosa con il token
}
```

Gli operatori & e &&, | e ||

Gli operatori & e | agiscono sui singoli bit degli operandi, restituendo un valore numerico i cui bit sono uno per uno il risultato dell'AND o dell'OR binario dei corrispondenti bit dei valori su cui operano. I due && e ||, invece, verificano il valore di verità dei due operandi (ognuno viene considerato solo come un VERO o FALSO) e restitui scono il risultato dell'AND o dell'OR dei due valori di verità riscontrati.

I tipi primitivi e gli oggetti

Se dovete utilizzare dei tipi primitivi (int, short, double, boolean, etc.) come oggetti, potete utilizzare le classi del linguaggio preposte a questa funzione, che sono Integer, Short, Double, Boolean, etc. Tutte hanno un costruttore che prende come parametro il tipo primitivo associato per impostarne il valore iniziale, e fanno parte del pacchetto implicito *java.lang*.

Java e PDF

Se dovete creare dei documenti PDF in Java potete utilizzare una di due librerie gratuite scaricabili da Internet: *www.lowagie.com/iText* oppure *www.etymon.com*.

Leggere un intero da una stringa

Se dovete interpretare un intero che ricevete in formato stringa, potete utilizzare il metodo statico parseInt dell'oggetto Integer, che si usa così:

```
String s1 = "1435";
int i = Integer.parseInt(s1);
// i = 1435
```

Eseguire comandi DOS

Il metodo exec() dell'oggetto Runtime vi consente di eseguire comandi di shell direttamente sul sistema operativo.
Ecco un esempio:

```
Runtime.getRuntime().exec(strComandoDos);
```

Caratteri UNICODE nelle stringhe

Per inserire un carattere UNICODE in una stringa Java, utilizzate la sequenza di escape \u seguite da quattro cifre esadecimali che indichino il carattere unicode da inserire… Per esempio,

```
String s1 = "\u0040";          // s1 = "@"
```

In questo modo si possono inserire nelle vostre stringhe anche caratteri di lingue orientali e medio-orientali.

Creare una data

La creazione di una data a partire da giorno, mese e anno si faceva tramite un costruttore dell'oggetto Date che prendeva anno (sottratto 1900), mese (sottratto 1) e giorno come parametri. Questo metodo è deprecato, e adesso si fa così

```
GregorianCalendar cal = new GregorianCalendar(year,month,day);
Date d = cal.getTime();
```

Leggere una response HTTP

Con questo semplice codice potete leggere una risposta HTTP come una qualunque stringa in forma di stream.

```
URL url = new URL(
        "http://www.federicomestrone.com");
BufferedReader in = new BufferedReader(new
InputStreamReader(url.openStream()));
String strin = in.readLine();
while ( strin != null ) {
        // Fai qualcosa con la stringa
        strin.readLine(); }
```

Servlet e JSP

Servlet e JSP sono praticamente la stessa cosa, in quanto entrambe rispondono ad una richiesta HTTP e consentono di inviare pagine HTML ad un browser. Le servlet però si scrivono con un editor Java, mentre le pagine JSP si compongono con editor HTML e sono quindi più orientate alla creazione della componente visuale. Dal punto di vista funzionale sono identiche e possono fare le stesse cose.

Impedire la costruzione di un oggetto

In un costruttore è possibile bloccare la costruzione di un oggetto lanciando un eccezione. In questo caso chi stava istanziando la classe dovrà gestire il problema e non riceverà alcuna istanza. Alternativamente si possono usare dei metodi statici di tipo factory.

Leggere risorse da un JAR

Se avete dei file di risorsa in un JAR, potete caricare tali risorse come stream utilizzando un metodo dell'oggetto Class, a patto che i file siano dentro il JAR e nella stessa directory del pacchetto della classe.

```
getClass().getResourceAsStream(nomefile);
```

Copia di array

Per copiare il contenuto di un array in un altro, invece di iterare e copiare elemento per elemento, è più semplice ed efficiente utilizzare il metodo arraycopy dell'oggetto System.

Impostare l'icona applicativa

Questo codice cambia l'icona utilizzata per rappresentare la vostra applicazione java in Windows:

```
myFrame.setIconImage(
        Toolkit.getDefaultToolkit().getImage(
                strFilename));
```

La variabile *myFrame* è un'istanza di *java.awt.Frame* o *javax.swing.JFrame*, mentre *strFilename* è una stringa con il nome del file d'icona. La classe Toolkit è nel pacchetto *java.awt*.

Eccezioni da catturare

Le eccezioni in java sono di due tipi: checked (devono per forza essere catturate in un blocco try…catch) e unchecked (vengono sollevate a runtime, non sono dichiarate e possono non essere catturate con try). Le eccezioni derivate da RuntimeException sono tutte unchecked ed includono ad esempio NullPointerException. Tutti gli Error, infine, che sono sempre Throwable e di cui un esempio è OutOfMemoryError, sono unchecked, mentre qualunque altro tipo di eccezione è applicativa, deve quindi essere dichiarata con la clausola throws ed intercettata con un try.

Eseguire codice allo shutdown

Se create un oggetto di tipo Thread e lo registrate come hook di shutdown, questo verrà eseguito allo shutdown della macchina virtuale, cioè prima della chiusura della JVM. Ecco il codice:

```
Runtime.getRuntime().addShutdownHook(
                threadClass);
```

Creare un'istanza a run-time

Anche senza conoscere, in fase di sviluppo, il tipo di oggetto che si deve creare è possibile comunque istanziare una classe con il codice seguente, dove strClass rappresenta il nome di una classe determinato a run-time.

```
Object obj = Class.forName(
                strClass).newInstance();
```

Viene invocato il costruttore privo di parametri.

Aprire un file da Java

Tramite il metodo exec, dell'oggetto Runtime, e il comando DOS cmd è possibile aprire un file con l'editor ad esso associato dal sistema operativo Windows. Il codice è il seguente

```
Runtime.getRuntime().exec("cmd /c "
                + filename);
```

Visual Basic

Come conoscere la directory di Windows?

La funzione riportata di seguito consente di recuperare il path completo relativo alla directory di installazione di Windows. Il valore ritornato dalla funzione corrisponde proprio alla stringa contenente l'informazione che cerchiamo.

```
Public Function GetWinDir() As String
Dim sWinDir As Stringx
Dim lReturn As Long 'Alocates memory for
                the string
sWinDir = String(255, Chr(0)) 'Tries to return
the Windows directory
lReturn = GetWindowsDirectory(sWinDir, 255)
                'Tests if an error occured
If lReturn = 0 Then GetWinDir = ""
Else
GetWinDir = Left(sWinDir, InStr(sWinDir,
                Chr(0)) - 1)
End If
End Function
```

Posso usare l'invio al posto del TAB?

Grazie al breve codice riportato, sarà possibile spostarsi all'interno dei cam-

pi di una Form utilizzando il tasto di invio invece del tasto tab il cui uso, benché frequente, può risultare innaturale per gli utenti meno esperti.

```
Private Sub Form_KeyPress(KeyAscii As Integer)
If KeyAscii = vbKeyReturn Then 'Catch Return Key
SendKeys "{TAB}" 'Send Tab which changes
                              active element on form
KeyAscii = 0 'Eat the Key to prevent a Beep
End If
End Sub
```

Qual è il massimo numero di controlli per una Form?

In Visual Basic 6.0 è possibile disporre un massimo di 255 controlli su una singola form.

Come posso salvare e leggere il contenuto di una textbox?

Per esemplificare la risposta a questa domanda, immaginiamo di avere una form con una textBox e due pulsanti. Settiamo la proprietà MultiLine della textBox a true. Associamo, ai due pulsanti della form, le due porzioni di codice riportate di seguito: il primo pulsante servirà per salvare il contenuto della textBox in un file chiamato “MyFile.txt", mentre attraverso la pressione del secondo pulsante, sarà possibile recuperare il contenuto dello stesso file e riproporlo nella textBox. Il codice che presentiamo funziona perfettamente così com'è, ma si presta ovviamente ad essere adattato per meglio venire in contro alle esigenze specifiche di un'applicazione.

```
Private Sub Command1_Click()
Open "MyFile.txt" For Output As #1
` This opens the file "MyFile.txt" in the current
directory for output.
Print #1, Text1.Text
` This prints all of the text in the textbox
                                          to the file
Close #1 ` This closes the file we just opened.
End Sub
Private Sub Command2_Click()
Text1.Text = "" ` Clear the textbox
Open "MyFile.txt" For Input As #1 ` Open
                       "MyFile.txt" so we can read it
Do Until EOF(1)
` Do the following code until we reach the end
                                          of the file
Line Input #1, temp$ ` Read in a line from
                                             the file
If Text1.Text <> "" Then
` If the textbox has text in it add the new text to it
Text1.Text = Text1.Text & vbCrLf & temp$
Else
` If the textbox is empty just stick our new
                                           text in it
Text1.Text = temp$
End If

Loop ` Loop back to the do as long as we
                         aren't at the end of the file
Close #1 ` Close the file
End Sub
```

Come posso utilizzare più colori in una textBox?

Una normale textBox supporta un singolo colore per volta, per ottenere l'effetto desiderato, abbiamo dunque due strade: utilizzare una RichTextBox o ricorrere ad una PictureBox. Per utilizzare il codice riportato di seguito, assicuriamoci di aver creato una form contenente due pulsanti, un RichTExtBox ed una PictureBox.
Lasciando inalterati i nomi proposti per default, il codice funzionerà così com'è.

```
Sub Command1_Click()
RichTextBox1.Text = ""
For x = 0 to 15
RichTextBox1.SelColor = QBColor(x)
` set the current color ( you don't 'need to
                      use the 16 color qb palette
` I'm using, I'm just using it to simplify color
                                       selection. )
RichTextBox1.SelText = "Color" & x & vbcrlf
` add our new text
Next x
End Sub

Sub Command2_Click()
Picture1.Cls ` Clears the PictureBox
For x = 0 To 15
Picture1.ForeColor = QBColor(x) ` sets the
                                      current color
Picture1.Print "Color" & x ` add our new text
Next x
End Sub
```

Come si possono aggiungere/rimuovere le voci di una listbox?

La prima cosa da tener presente, quando si lavora con le listbox, è che gli indici utilizzati partono tutti da zero, cioè una listbox contenente cinque elementi, li avrà indicizzati da 0 a 4.
Nel momento in cui si aggiunge una voce, se non si specifica la posizione, essa sarà inserita alla fine della lista.
Ecco come aggiungere una voce ad una listBox utilizzando il metodo AddItem:

```
List1.Clear ` Clears the listbox
List1.AddItem ("Primo") ` Aggiunge la voce
                 "Primo" alla listbox con indice 0
List1.AddItem ("Secondo") 'Aggiunge la voce
               "Secondo" alla listbox con indice 1
List1.AddItem ("Terzo", 0) 'Aggiunge la voce
                 "Terzo" alla listbox con indice 0
```

Avendo noi specificato l'indice per il terzo valore inserito, l'ordine della listBox sarà: Terzo, Primo, Secondo.
Per rimuovere una voce da una listBox è necessario conoscere l'indice corrispondente alla voce stessa:

```
List1.RemoveItem 0 ` Rimuove la voce avente
                                      come indice 0
```

Come visto, se si conosce l'indice della voce che vogliamo eliminare, l'operazione di rimozione è quanto mai banale. Ma cosa accade se non conosciamo questo dato?
La porzione di codice seguente si occupa proprio di risolvere questo problema, ciclando sulla listbox alla ricerca della voce da eliminare e cancellandola.

```
List1.Clear ` Clear the listbox
For x = 0 To 8
List1.AddItem ("Foo" & 0) ` Add some items
                                     to the listbox
Next x
For x = 0 To List1.ListCount – 1 ` Loop through list
` remember index 0 counts as an item so the
                       ListCount will always be one
` higher then the top boundary. If you have 3
                          items they are 0-2 not 1-3
If List1.List(x) = "Foo6" Then ` Check this
                                         items data
List1.RemoveItem x ` Data matches, so
                                    remove the item
Exit Sub ` Exit the procedure because we
                             found what we wanted
End If

Next x
```

Tips&Tricks

I trucchi del mestiere

La rubrica raccoglie trucchi e piccoli pezzi di codice che solitamente non trovano posto nei manuali, ma sono frutto dell'esperienza di chi programma. Alcuni trucchi sono proposti dalla Redazione, altri provengono da una ricerca sulla Rete delle Reti, altri ancora ci giungono dai lettori. Chi vuole contribuire potrà inviarci i suoi tips&tricks preferiti che, una volta scelti, verranno pubblicati nella rubrica. Il codice completo dei tips lo trovate nel CD allegato nella directory \tips\.

Visual Basic

Come richiamare la finestra "Trova" del menù Start

Il tips proposto, consente di richiamare la funzione "*Trova*" di Windows. Come è facile intuire dal codice, il tips utilizza l'API di sistema *ShellExecute*. Tale API, usata opportunamente, consente di eseguire molte operazioni prettamente di sistema.

Come richiamare la finestra delle proprietà di un file

Spesso è fondamentale sapere la data di creazione di un particolare file, lo spazio occupato su disco e diverse altre proprietà. Il tips consente di ricavare tutte le informazioni relative alle proprietà di un file, così come normalmente accade cliccando con il tasto destro del mouse e richiamando la voce *Proprietà*.

Come invocare un report Access da Visual Basic

E' risaputo che Microsoft Access offre un sistema di report molto più all'avanguardia rispetto a Visual Basic 6.0. Proprio per questo motivo, il codice indicato può essere molto utile se si decide di avviare una stampa di un report Access direttamente da un'applicazione VB.

Come ottenere info sulla Barra delle Applicazioni di Windows

In alcune applicazioni potrebbe rendersi necessario verificare alcune impostazioni della barra delle applicazioni di Windows; per esempio si potrebbe voler verificare se le opzioni di *"Nascondi automaticamente"* e/o *"Sempre in primo piano"* sono attive o meno.

Come inviare una stringa di testo ad una shell DOS

Il tips consente di utilizzare la clipboard per "immagazzinare" e successivamente inviare, una stringa di testo alla shell DOS.
In questo esempio, viene simulato il passaggio della stringa *"dir *.*"* e la pressione del tasto invio; ricordiamo, se ce ne fosse bisogno, che la stringa *"dir *.*"* permette di reperire l'elenco di tutti i file presenti nella directory corrente.

ASP

Come ottenere il contenuto di una cartella

Per analizzare il contenuto di una cartella del server attraverso un documento ASP, è necessario ricorrere all'oggetto *ActiveX Scripting.FileSystemObject*. La funzione qui di seguito riportata, prende in ingresso il nome di una cartella, e quindi ne analizza il contenuto restituendo un array di stringhe in cui ogni elemento rappresenta il nome di uno dei file contenuti nella cartella (il codice è JScript).
Trucco fornito da *Carlo Pelliccia*

Come inviare e-mail senza il componente CDONTS

Non tutti i Web server che supportano ASP sono abilitati all'invio di e-mail tramite il noto componente CDONTS. Un'alternativa parecchio diffusa è rappresentata infatti dall'ActiveX AspEmail di Persist Software (*http://www.aspemail.com/*). Vediamo come inviare una e-mail tramite questo componente (il codice è VBScript).
Trucco fornito da *Carlo Pelliccia*

Come connettersi ad un DB con ASP.NET

La connessione ad un DB, utilizzando ASP.NET, rimane abbastanza simile nel caso in cui si utilizza ASP. È tuttavia utile riproporre il nuovo metodo di connessione; in questo esempio si realizza la connessione a un database Microsoft Access.

Come evidenziare una parola all'interno di un testo

Con questo script vogliamo mostrare come, utilizzando la funzione *Replace* combinata ad un ciclo, sia possibile creare un'opportuna funzione di evidenziazione parole; in questo contesto faremo uso del tag Span e dei CSS.

Java

Come interagire con un database Access

Java realizza le connessioni ai database attraverso il compo-

nente noto come JDBC. Poiché Java è multipiattaforma, tale insieme di API fornisce funzionalità generiche per l'accesso alle fonti condivise, indipendentemente dal loro formato. Il trucco è abbastanza semplice: JDBC fa da "tramite" tra un'applicazione Java ed il DBMS che si intende sfruttare, spesso attraversando anche altri collegamenti interni. Per testare la connessione con gli archivi di Access, realizzate un archivio chiamato *miodatabase.mdb*, e al suo interno create una tabella nominata *Persone*, suddivisa in quattro campi:

- *ID*, un banale contatore usato come chiave primaria.
- *Nome*, di tipo testuale.
- *Cognome*, di tipo testuale.
- *Indirizzo*, di tipo testuale.

Quindi popolate la tabella con qualche dato arbitrario.
A questo punto è necessario far riconoscere al sistema operativo il database, in modo che possa essere introdotto tra le fonti di dati ODBC gestite. Per far ciò, è necessario configurare un DSN. L'operazione è semplice:

1. Dal "pannello di controllo" di Windows accedete alla voce *"origine dati ODBC"*.
2. Selezionate "*aggiungi*" nella scheda "*DSN utente*".
3. Scegliete, dall'elenco presentato, il driver di Access, identificato solitamente dalla dicitura "Microsoft Access Driver (*.mdb)".
4. Nella maschera ora presentata, immettete i dati necessari per la creazione del DSN. Usate il nome *MioDatabase*, inserite una descrizione a piacere, e con il pulsante "*seleziona*" collegate il DSN al file *miodatabase.mdb* precedentemente creato.
5. Confermate ogni operazione effettuata, salvando così il DSN utente creato.

Ora la base di dati è ufficialmente riconosciuta dal sistema operativo, JDBC può accedervi sfruttando, come tramite, il driver ODBC del sistema. In casi come questo, infatti, si parla di ponte JDBC-ODBC. Le classi Java che permettono l'accesso ai database sono contenute nel package *java.sql*. Sul CD è presente un esempio in grado di sfruttare il database di Access appena registrato nel sistema.
Trucco fornito da *Carlo Pelliccia*

Come eseguire clip sonori in Java

Utilizzando le Java Sound API è possibile caricare ed eseguire dei clip sonori in vari formati all'interno delle proprie applicazioni. La classe *Sounds*, qui sotto, mostra come fare. Può essere utilizzata creandone un'istanza all'interno del proprio programma, passando al costruttore un array di stringhe rappresentanti i nomi dei diversi file sonori da caricare (ad esempio *"nomeClip.wav"*). Diventa quindi possibile eseguire un determinato clip lanciando il metodo *play(numeroClip)*. Nel momento in cui i clip sonori caricati non risultino più necessari, è possibile chiuderli tutti insieme utilizzando il metodo *closeClips()*.
Trucco fornito da *Carlo Pelliccia*

Come creare uno splash screen

Molte applicazioni, al loro avvio, mostrano uno "splash screen", ossia una finestra contenente un logo ed alcune informazioni, che intrattengono l'utente mentre il software viene caricato e predisposto per l'utilizzo. Applicazioni che ne fanno uso, ad esempio, sono Netscape, Word e Outlook Express.
Uno splash screen si distingue da una normale finestra per via del fatto che non contiene elementi come la barra del titolo, i contorni e il pulsante di chiusura. Uno splash screen, inoltre, scompare automaticamente a caricamento completato, oppure dopo un determinato numero di secondi. In Java è possibile replicare tale comportamento usando la classe *Window* dell'AWT.
Non è possibile usare frame, poiché questa classe fornisce una finestra già popolata con diversi elementi. L'esempio, mostra l'immagine "*splash.jpg*", in qualità di splash screen, per dieci secondi.
Trucco fornito da *Carlo Pelliccia*

Sviluppare OGGETTI COM

Pocket PC

Impararare a costruire oggetti COM utilizzando tecniche standard è fondamentale per la implementazione delle nostre applicazioni. Vediamo come possiamo estendere tali concetti anche per Windows CE con dispositivi Pocket PC.

File sul CD
\soft\codice\FileAccessServer.zip

In questo articolo ci occuperemo dello sviluppo di un componente COM utilizzando l'ambiente Embedded Visual C++ 3.0. Prima di tutto, illustreremo lo scenario applicativo, ovvero spiegheremo quale sarà l'insieme di funzionalità che l'oggetto dovrà implementare. Una volta costruito l'oggetto COM, impareremo ad inserirlo in un'applicazione e ne invocheremo i servizi esportati all'esterno.

LO SCENARIO APPLICATIVO

Il nostro scopo è quello di sviluppare un oggetto che si interfaccia con il File System di Windows CE. Esso dovrà consentirci non solo di scrivere dati su file ma anche di recuperarli con una classica operazione di lettura. Una volta riusciti nel nostro intento, avremo fornito alla nostra applicazione una prima tecnica per archiviare i nostri dati. Altre tecniche di memorizzazione dei dati sono riconducibili, ad esempio, ai data base relazionati. Per quanto riguarda quest'ultimi, possiamo affermare che anche nelle applicazioni di Mobile Computing abbiamo a disposizione data base che rispondono ai nomi di Pocket Access oppure SQL Server CE, per quanto riguarda Microsoft, ma anche prodotti come Oracle Light. Supponiamo di volere monitorare i movimenti di magazzino di un fornitore. I movimenti sono rappresentati da ordini di prodotti da parte dei propri clienti. Il magazziniere dovrà leggere du supporto cartaceo la descrizione degli ordini e memorizzarli sul dispositivo palmare. A fine giornata, i dati saranno visualizzati in una maschera della applicazione ed, eventualemnte, soggetti ad ulteriore elaborazione. Per quanto riguarda la struttura degli ordini precisiamo che si tratta di una lista di prodotti richiesti da un determinato cliente e in una determinata quantità.

Fig. 1: Creiamo un nuovo progetto con ATL.

COSTRUZIONE DEL PROGETTO COM ATL

Entriamo subito nel vivo nel progetto con la costruzione del del componente COM utilizzando il modello ATL di Microsoft eMbedded Visual C++. È appena il caso di precisare che ATL rappresenta un insieme di classi C++ con modelli che permettono di costruire componenti COM efficienti, senza dover eseguire operazioni complicate. Inoltre, i componenti creati con ATL sono piccoli, veloci e facili da manutenere. Per creare un oggetto COM mediante ATL è necessario effettuare i seguenti passi:

1. Aprire *eMbedded Visual C++ 3.0* e selezionare *"New.."* dal Menù *File*.

2. Nella Dialog Box *"New"* selezionare la scheda *"Projects"* e scegliere *"WCE ATL COM AppWizard"*. Digitare *FileAccessServer* nel campo *"project name"*. Inoltre, selezionare le CPUs per le quali si

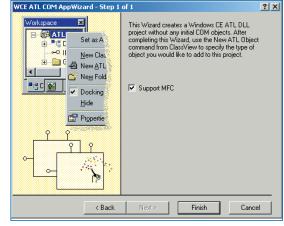



Fig. 2: Il wizard ATL COM.

desidera rendere disponibile l'applicazione e cliccare *OK*.

3. A questo punto il Wizard chiederà se includere il supporto per MFC. Selezioniamo la casella poichè prevediamo l'utilizzo di classi MFC nel nostro progetto.

4. Premere il tasto *Finish*.

5. Premere il tasto destro del mouse sulla radice *"FileAccessServer"* del *Project Explorer* e nel menù contestuale selezionare *"New ATL Object"*.

6. Nella finestra di dialogo *"ATL Object Wizard"* selezionare *"Simple Object"* nella categoria *"Object"*.

7. A questo punto il Wizard chiederà informazioni che riguardano le classi C++ da generare e le interfaccie COM. In particolare, nella scheda *"Attributes"* manteniamo le impostazioni di default. Nella scheda *"Names"*, nel campo *"Short Name"*, digitiamo *FileAccess* nel contempo gli altri campi verranno verranno riempiti in modo automantico (Fig. 3).

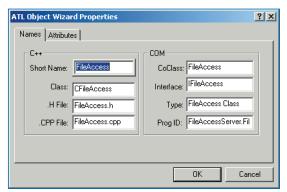



Fig. 3: Proprietà dell'oggetto ATL.

8. La pressione del tasto *OK* comporterà la visualizzazione nel *Tree View* del *Project Explorer* della classe *CFileAccess* e l'interfaccia *IFileAccess*. Quest'ultima rappresenterà la dichiarazione delle funzionalità del componente, mentre la classe *CFileAccess* rappresenta l'effettiva implementazione.

AGGIUNTA DI METODI E PROPRIETÀ ALL'INTERFACCIA DEL COMPONENTE

Creata la classe ed una interfaccia per il componente in questione, dobbiamo aggiungervi dei metodi ed, eventualmente, delle proprietà. Prima di tutto aggiungiamo una proprietà, il cui valore attuale consentirà all'oggetto di capire quale sia il nome del file su cui scrivere o da cui leggere i dati. Per aggiungere la proprietà seguiamo i seguenti passi:

1. In *ClassView* fare click con il pulsante destro del mouse sulla interfaccia *IFileAccess*.

2. Scegliere *"Add Property"* dal menù di scelta rapida.

3. Selezionare *HRESULT* nella casella *"Return Type"*; selezionare *BSTR* nella casella *"Property Type"*.

4. Digitare *FileName* nella casella *"Property Name"*, nel contempo, nella casella *Implementation* della dialog *"Add Property"*, possiamo osservare le firme *MIDL* per le funzioni *get_FileName()* e *put_FileName()*. Queste ultime due funzioni permettono, rispettivamente, di recuperare il valore attuale della property e di modificarla.

5. Premere *Ok* per creare le funzioni che ci permetteranno di accedere alla proprietà del componente.

La procedura che permetterà di rendere effettiva l'aggiunta della proprietà, consisterà nell'aggiungere alla classe di implementazione *CFileAccess* una variabile che conterrà effettivamente i dati. Sarà poi nostra cura implementare le funzioni *get* e *put* per inviare e ricevere i dati dalla variabile. Per implementare la proprietà *FileName* seguiamo i seguenti passi:

1. Aggiungiamo alla classe *CFileAccess* una variabile protetta di nome *m_fileName* di tipo *BSTR;*

2. Inizializziamo la variabile membro nel suo costruttore alla stringa vuota; quindi il costruttore avrà avrà la seguente struttura:

```
CFileAccess()
 { m_FileName = ::SysAllocString(_T(""));}
```

Mentre il distruttore con l'ausilio della funzione *SysFreeString* permetterà di liberare lo spazio assegnato all'oggetto *BSTR* che rappresenta un puntatore ad una stringa con caratteri estesi (ovvero di tipo **wchar_t*):

```
 ~CFileAccess()
 { ::SysFreeString(m_FileName);}
```

3. In *ClassView* espandere il nodo dell'interfaccia *IFileAccess* al di sotto della classe *CFileAccess*; in tal modo sarà possibile individuare le funzioni *get_FileName()* e *put_FileName*. Implementiamo le funzioni in base al codice seguente:

```
STDMETHODIMP CFileAccess::get_FileName(BSTR
```

Pocket PC

Sviluppare
Oggetti COM

Windows CE

Rispetto all'analogo Wizard per la versione Desktop, dobbiamo notare che nella versione per Windows CE non ci viene chiesta la tipologia di server (DLL, EXE, Service), il supporto per MTS oppure proxy/stub merging: tali opzioni non sono supportate in Windows CE.

Pocket PC

Sviluppare
Oggetti COM

Compilazione

Per poter compilare il componente COM senza problemi, bisogna che sia settata un importante proprietà del progetto. È necessario, infatti, pemettere la multipla definizione di simboli all'interno della omonima tabella. Se non si forzasse tale proprietà risulterebbe in un errore di compilazione. Ecco la procedura da seguire:

1. Dal menù *Project* selezionare *Settings...* verrà aperta la maschera a schede *Project Setting.*

2. Selezionare la scheda link e nella casella *Force Symbol References* scrivere *MULTIPLE.*

3. Nella casella *Project Options* ricercare il contenuto *"include: MULTIPLE"* e sostituire la stringa include con force.

4. Premere *Ok* per rendere effettive le modifiche alle proprietà del progetto.

```
*pVal)
{  *pVal = ::SysAllocString(m_FileName);
   if(*pVal == NULL)
   { return E_FAIL }
   return S_OK; }
```

Nel codice sopra riportato possiamo fare le seguenti considerazioni:

- Alla funzione *get_FileName* viene fornito un puntatore ad un tipo *BSTR*. Tale parametro conterrà, alla fine, il valore corrente della property del componente.
- Nella prima linea di codice , viene copiato il contenuto della variabile *m_FileName* in **pVal* con l'ausilio dell'API di Windows CE *SysAllocString*. Quest'ultima funzione ritorna un *BSTR* che avrà valore *NULL* se non vi è sufficiente memoria oppure se il suo agomento è *NULL*. Nel caso vi fosse un errore di questo tipo, la funzione ritorna un tipo *HRESULT* appropriato. In particolare, nel caso di un errore la funzione restituisce *E_FAIL* a indicare una situazione di errore.

```
STDMETHODIMP CFileAccess::put_FileName(BSTR newVal)
{
   m_FileName = ::SysAllocString(newVal);
   if(*m_FileName == NULL)
   { return E_FAIL }
   return S_OK; }
```

Per la funzione *put_FileName* possiamo fare considerazioni simili alle precedenti. In sintesi, eseguiamo la copia del contenuto del variabile *newVal* passata come argomento, nella variabile *m_FileName*. Anche in questo caso, poniamo attenzione a che l'operazione di allocazione della memoria sia andata a buon fine. Consigliamo di effettuare una compilazione del componente di volta in volta che viene fatta una modifica.

LOGICA DI SCRITTURA SU FILE

Dopo aver aggiunto al codice del componente, le funzioni per accedere in lettura e in scrittura alla sua proprietà *m_FileName*, possiamo aggiungere la logica di scrittura di dati in un *File*. Impareremo a gestire gli oggetti del File System di Windows CE unitamente all'utilizzo delle API per la gestione dei file. Prima di tutto è necessario stabilire un formato nella scrittura dei dati. Una volta stabilito tale formato sarà poi possibile anche recuperare i dati dal file di memorizzazione. Le informazioni che il file dovrà contenere possiamo così definirle:

- L'informazione sul numero di ordini.
- La lista degli ordini.
- Un ordine è caratterizzato dalle seguenti informazioni:
  1. Il numero di prodotti costituenti l'ordine.
  2. Il cliente che richiede i prodotti.
  3. Un insieme di coppie (nome prodotto, quantità).

Definite le informazioni che il file dovrà contenere, defiamo la struttura del file, ovvero il cosiddetto *"tracciato record"*. Un tracciato record rappresenta un modo per assemblare un insieme di informazioni. Intanto tali informazioni avranno un senso in quanto abbiamo stabilito un esatto ordine e un preciso formato di inserimento nel file. Se l'inserimento di tali informazioni non rispettasse le regole del tracciato record, esse non potranno essere recuperate in fase di lettura. Allo stesso modo, se l'applicazione che legge il file per recuperare i dati non rispettasse le regole, allora non riuscirebbe a ricostruire i dati originali. Andiamo quindi a stabilire le regole del tracciato record in modo intuitivo:

```
[Numero_Ordini(2Byte)] {Lista_Ordini}
{Lista_Ordini}=[Numero_Prodotti_Ordine(2Byte)]
                             [Cliente]{Lista_Prodotti}
{Cliente} = [Numero_Byte_Stringa_Nome_Cliente(
                              1 Byte)}[Nome_Cliente]
{Lista_Prodotti} = [Numero_Prodotti_in_ordine] {
                                              Ordine}
[Ordine] = [Numero_Byte_Stringa_Nome_Prodotto]
                    [Nome_Prodotto][Quantità(2 Byte)]
```

Nella precedente rappresentazione, gli oggetti inseriti tra parentesi quadre sono entità singole, eventualmente, seguite dal numero di byte usati per rappresentarli e inseriti fra parentesi tonde. Gli oggetti tra parentesi graffe si riferiscono invece a liste di oggetti. Un possibile esempio del tracciato record prima specificato potrebbe essere il seguente:

```
Esempio Tracciato Record.
1 2  5 PIPPO 2 PC 32 7 SCANNER 25
```

Scrivere in un file tale stringa, equivale a memorizzare un solo ordine (l'intero 1); l'ordine, emesso dal cliente PIPPO, è composto da 2 tipologie di prodotti: computer (32 unità) e scanner (25 unità). Notiamo che ogni stringa sia preceduta da un Byte che rappresenta la sua lunghezza. Per cui *"5 PIPPO"* consente alla applicazione che legge il file che leggerà una stringa di 5 caratteri. Analoghe considerazioni valgono per le altre stringhe di descrizione, ovvero *"2 PC"* e *"7 SCANNER"*. L'esempio precedente rappresenta, naturalmente, un modello ad alto livello dei dati. Nel File System di Windows CE la rappresentazione è ovviamente a livello di insieme di Byte.

Pocket PC

Sviluppare
Oggetti COM

Inoltre, non esistono spazi tra i campi. Aggiunta dei Metodi alla interfaccia del componente. Andiamo, a questo punto, ad aggiungere al componente COM le nostra funzione di creazione del tracciato record.
Seguire il seguente insieme di passi per aggiungere una funzione al componente:

1. In *ClassView* premere il tasto destro del mouse sulla interfaccia *IFileAccess.*

2. Nel menù di scelta rapida selezionare *"Add Method"*. Viene visualizzata la finestra di dialogo *Add Method to Interface.*

3. Nella casella *Return Type,* selezionare *HRESULT.*

4. Nella casella *Method Name,* digitare *CreaTracciatoRecord.*

5. Nella casella *Parameters,* aggiungere il seguente codice:

```
[in] const BSTR path,[in] WORD wNumOrdini
```

Nella casella *Implementation* della *Dialog* viene mostrato un listato *MIDL* completo della firma del metodo in base ai dati immessi. Il parametro di ingresso alla funzione rappresenta il percorso nel File System di Windows CE, in cui vogliamo memorizzare il file del tracciato record.
Il secondo parametro di ingresso rapresenta il numero di ordini che vogliamo scrivere nel file. Una possibile implementazione è la seguente:

```
STDMETHODIMP CFileAccess::CreaTracciatoRecords(
                        const BSTR path, WORD wNumOrdini)
{
  CString strPath = path;
  CString strFileName= m_FileName;
  CString strPathCompletoFile= strPath + strFileName;
      HANDLE hFile = CreateFile(strPathCompletoFile,
          GENERIC_WRITE, NULL, NULL, CREATE_ALWAYS,
                                          NULL, NULL);
   if (hFile == INVALID_HANDLE_VALUE)
    { return E_FAIL; }
    DWORD dwWritten;
    bool bRet = WriteFile(hFile,&wNumOrdini,sizeof(
                          WORD),&dwWritten,NULL);
    return S_OK; }
```

Dal codice possiamo notare che, una volta costruito il path completo del File e memorizzato il suo valore nella variabile *strPathCompleto,* tramite la funzione *CreateFile* si costruisce un file con il nome e il percorso specificato nel primo argomento. *hFile* di tipo *HANDLE,* rappresenta una variabile membro della classe di implementazione del componente, la classe *CFileAccess.*
La variabile è stata aggiunta per memorizzare l'handle del file creato dalla funzione *CreateFile.* Infatti, quest'ultima funzione di Windows CE, restituisce un *Handle* al file oppure *INVALID_HANDLE_VALUE* se l'operazione di creazione non è andata a buon fine. In Tab. 1 diamo una breve documentazione della funzione con i suoi argomenti e il valore di ritorno.

Funzione	CreateFile
LPCTSTR name	Nome del file da aprire
DWORD accessMode	Modalità di apertura : *Read/Write*
DWORD shareMode	Modalità in cui il file dovrebbe essere condiviso
LPSECURITY_ ATTRIBUTES securityAttributes	Parametro non supportato in Windows CE, ha sempre valore *NULL*
DWORD create	Modalità di creazione del file
DWORD attributes	Maschera di bits per gli attributi del file
HANDLE templateFile	Non supportato in Windows CE, per cui viene passato con il valore *NULL*
HANDLE Valore di uscita	HANDLE al file oppure *INVALID_HANDLE_VALUE* se interviene un errore

Tab. 1: Parametri della Funzione CreateFile.

Se l'operazione di creazione del file è andata a buon fine scriviamo 2 Byte rappresentati il numero di ordini. Quest'ultima operazione è stata effettuata tramite la funzione *WriteFile.*
In Tab. 2 sono descritti i parametri di ingresso della funzione e il valore di ritorno. Per un ulteriore approfondimento rimandiamo alla documentazione MSDN.

Funzione	WriteFile
HANDLE FileHandle	*Handle* del File creato da *CreateFile*
*CONST VOID *buffer*	I dati da scrivere nel file
DWORD bytesToWrite	Il numero di byte da scrivere
LPDWORD bytesWritten	Puntatore ad una *DWORD* che ritorna il numero di byte effettivamente scritti
LPOVERLAPPED overlapped	Non supportata in Windows CE; passare sempre valore *NULL*
BOOL Valore di ritorno	*TRUE* per successo, *FALSE* per fallimento

Tab. 2: Parametri della Funzione WriteFile

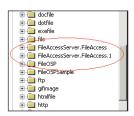
Aggiungiamo le altre funzioni alla interfaccia del componente COM. Il nome del successivo metodo sarà *InserisciCliente* con la seguente firma *MIDL:*

Compilazione del codice

È il caso di evidenziare, che se l'operazione di compilazione del codice del componente è andata buon fine, Embedde Visual C++ 3.0 verificherà se sul dispositivo sono presenti tutte le DLL richieste per il buon funzionamento del componente COM. In particolare, il Server COM richiede la presenza del modulo ATLCE211.DLL. Nel caso quest'ultima DLL non fosse presente l'ambiente si preoccuperebbe di effettuarne il download sul Pocket PC e di registrarla in automatico.

Sviluppare
Oggetti COM

Registrazione componente COM

eMbedded Visual C++ effettuerà anche il download e la registrazione del nostro Server COM. Per verificare la registrazione con successo del componente possiamo utilizzare il tool *"Windows CE Remote Registry Editor"* nel menù *Tool* e cercare nella folder di registro *"HKEY_ CLASSES_ROOT"* la chiave *FileAccessServer.FileAccess.*

```
[in] const BSTR Cliente,[in] WORD wNumProdotti
```

- Il primo parametro rappresenta il nome del cliente che ha effettuato l'ordine.
- Il secondo parametro rappresenta il numero di prodotti che compongono il suo ordine.

Tali informazioni saranno anch'esse scritte nel file per comporre il tracciato record. Ecco una sua implementazione:

```
STDMETHODIMP CFileAccess::InserisciCliente(
                    const BSTR Cliente, WORD wNumProdotti)
{
    DWORD bytesWritten;
    CString strCliente = Cliente;
    BYTE length=strCliente.GetLength() ;
    TCHAR buffer[255];
    _tcscpy(buffer, strCliente);
    /*scrivo la lunghezza della stringa strCliente*/
    bRet=WriteFile(hFile,&length,sizeof(BYTE),
                              &bytesWritten,NULL);
    /*scrivo il nome del cliente*/
    bRet=bRet && WriteFile(hFile,&buffer,sizeof(
                    TCHAR)*length,&bytesWritten,NULL);
    /*scrivo 2 byte rappresentanti il numero di prodotti
                                          nell'ordine*/
    bRet=bRet && WriteFile(hFile,&wProdotti,sizeof(
                         WORD),&bytesWritten,NULL);
    return S_OK; }
```

Tramite questa funzione si riesce a costruire le informazioni preliminari che riguardano un ordine di un cliente. Inoltre, le suddette informazioni sono scritte secondo le regole specificate nel tracciato record.

AGGIUNGIAMO UN NUOVO PRODOTTO ALLA LISTA

Andiamo a creare il metodo che permette di aggiungere al tracciato record le informazioni relative ad un prodotto richiesto dal cliente corrente. All'interfaccia del componente, aggiungiamo un metodo con il nome *InserisciProdotto* con la seguente firma MIDL:

```
[in] const BSTR Prodotto,[in] WORD wQuantita
```

- Il primo parametro rappresenta la descrizione del prodotto richiesto dal cliente
- Il secondo parametro rappresenta la quantità richiesta del prodotto.

Ecco la sua implementazione:

```
STDMETHODIMP CFileAccess::InserisciProdotto(
                    const BSTR Prodotto, WORD wQuantita)
{
    DWORD bytesWritten;
    CString strProdotto = Prodotto;
    BYTE length = strProdotto.GetLength() ;
    TCHAR buffer[255];
    _tcscpy(buffer,strProdotto);
    /*scrivo la lunghezza della stringa strProdotto*/
    bRet=WriteFile(hFile,&length,sizeof(
                         BYTE),&bytesWritten,NULL);
    /*scrivo il nome del Prodotto*/
    bRet=bRet && WriteFile(hFile,&buffer,sizeof(
                    TCHAR)*length,&bytesWritten,NULL);
    /*scrivo i 2 byte rappresentanti la quantità del
                                  prodotto richiesto*/
    bRet=bRet && WriteFile(hFile,&wQuantita,
                    sizeof(WORD),&bytesWritten,NULL);
    return S_OK;
}
```

Il precedente metodo ha una struttura molto simile al precedente. Inoltre, i commenti aiutano a capirne la logica. Prima di scrivere i metodi che permettono di leggere i dati dal file del tracciaro record, dobbiamo avere la possibilità di rilasciare l'handle del file creato. A tale scopo si aggiunga all'interfaccia il metodo con il nome *ChiudiTracciatoRecord* senza argomenti. Una sua implementazione è la seguente:

```
STDMETHODIMP CFileAccess::ChiudiTracciatoRecord()
{ CloseHandle(hFile);
  return S_OK; }
```

Viene richiamata la funzione *CloseHandle* per chiudere l'handle associato al file. Ovvero rilasciamo la risorsa prima allocata. La funzione *CloseHandle* di Windows CE ritorna *TRUE* se l'operazione è andata a buon fine, *FALSE* altrimenti.

Funzione	ReadFile
HANDLE FileHandle	*Handle* del File creato da *CreateFile*
LPVOID buffer	Buffer in cui mantenere i dati letti
DWORD requestedBytes	Il numero di byte che si vogliono leggere
LPDWORD actualBytes	Il numero di bytes effettivaemnte presenti nel buffer
LPOVERLAPPED overlapped	Non supportata in Windows CE; passare sempre valore *NULL*
BOOL Valore di ritorno	*TRUE* per successo, *FALSE* per fallimento

Tab. 3: **Parametri della Funzione ReadFile.**

CONCLUSIONI

Lo spazio a nostra disposizione termina qui, in un prossimo articolo analizzeremo come avviene il recupero dei dati dal file, nonché come effettuare il test del componente.

Ing. Elmiro Tavolaro

BIOSInfo

RECUPERARE LE INFORMAZIONI DA VB

Sistema

Il BIOS (Basic Input Output System) è l'insieme di istruzioni base che permettono al computer di eseguire la procedura di accensione, di riconoscere l'hardware installato e di caricare il sistema operativo da Floppy o da Hard Disk. Vediamo come accedervi mediante Visual Basic.

File sul CD
\soft\codice\BiosInfo.zip

Il BIOS, acronimo di *Basic Input Output System* (Sistema di base per l'immissione /emissione dati), è uno speciale software residente in una memoria di tipo PROM o Flash ROM (nelle schede più recenti). Questo chip contiene una serie di istruzioni per il microprocessore del PC con un codice che viene eseguito automaticamente all'accensione del computer. Il BIOS si incarica di eseguire una procedura di autodiagnostica *(POST, Power On Self Test)* che procede a tutta una serie di controlli e verifiche sulla memoria RAM, tastiera, processore, drive, disco fisso, porte di comunicazione e che permette, infine, di caricare il sistema operativo da floppy disk o da hard disk. Affinché il BIOS possa eseguire correttamente il suo compito, necessita di conoscere l'hardware installato nel personal computer e la sua configurazione. Tutti i parametri di configurazione e setup necessari al funzionamento del BIOS e del computer, sono registrati su di una memoria chiamata RAM CMOS (una memoria a bassissimo consumo di appena 64 Kb) che rimane costantemente alimentata. Quando il PC è spento, tale alimentazione viene garantita da una piccola batteria tampone presente sulla scheda madre. Ne consegue che un esaurimento di questa batteria provocherà la perdita dei dati contenuti nella RAM CMOS, con inevitabili problemi all' avvio della macchina e la necessità di riconfigurare il setup dopo aver sostituito la batteria. A tale proposito, è buona norma stampare tutte le impostazioni originali del BIOS setup per poterle ripristinare in caso di difficoltà. Se la scheda madre è equipaggiata con un BIOS su Flash ROM riprogrammabile, esiste la possibilità di aggiornarlo periodicamente con nuove versioni sviluppate dai produttori di schede. Il software è prelevabile via Internet presso il sito del produttore. L'aggiornamento del BIOS offre sicuramente dei vantaggi, ma può comunque comportare dei problemi. Una improvvisa assenza della tensione di alimentazione durante l'aggiornamento o un inadatto file di upgrade possono produrre conseguenze molto serie. In questi casi ci troveremmo ad avere una scheda madre priva di BIOS e quindi assolutamente inutilizzabile. Se ci dovessimo malauguratamente trovare in una situazione di questo tipo, prima di dissaldare il chip e riprogrammarlo con un programmatore esterno con costi abbastanza alti, paragonabili al valore della motherboard, una possibile soluzione potrebbe essere quella di eliminare tutti i collegamenti dalla scheda madre alle periferiche lasciando attaccato il solo floppy disk e sostituire la scheda video con una vecchia e cara scheda ISO. Nel BIOS è infatti presente un segmento d'area predisposto a questo tipo di evenienza. Inseriamo nel drive un floppy bootabile con su caricato tutto il necessario per l'aggiornamento del BIOS e riaccendiamo il pc. Dovrebbe effettuare il boot da floppy e quindi, con un pò di fortuna sarà possibile recuperare il tutto. Upgradato il BIOS, risistemare tutti i collegamenti della scheda madre, sostituire la scheda video e riavviare il computer incrociando le dita! Sarebbe comunque opportuno, per evitare cataclismi del genere, prima di procedere all'aggiornamento, fare delle copie di sicurezza del BIOS di serie utilizzando l'apposita utilità software fornita con la scheda madre onde evitare che una volta sostituito venga eliminato per sempre. Esiste comunque, sulla maggior parte delle schede madri, un jumper che opportunamente impostato, consente o meno di sovrascrivere sul chip. Ma cosa c'è scritto all'interno del BIOS presente sulla nostra scheda madre? Attraverso l'applicativo che andremo a creare, ne recupereremo tutte le informazioni principali. Tra questi abbiamo il nome, la descrizione, la data dell'ultimo aggiornamento, la versione, ecc. Nella scrittura del codice, ci ritorna utile l'utilizzo della libreria dei tipi di *Microsoft Windows Management Instrumentation Scripting WBEMDISP.TLB.*

MICROSOFT WINDOWS MANAGEMENT INSTRUMENTATION SCRIPTING (WMI)

WMI è un'infrastruttura standardizzata per il monitoraggio e la gestione delle risorse del sistema. WMI consente agli amministratori di sistema di

monitorare e controllare il sistema tramite scripting e applicazioni di terze parti.
La combinazione delle estensioni WMI per WDM *(Windows Driver Model)* e dei servizi sviluppati in conformità con gli standard Web-Based Enterprise Management *(WBEM)* consente di semplificare la strumentazione e offrire un accesso ampio e coerente ai dati di gestione. Queste estensioni sono già integrate nel kernel dei sistemi operativi Windows 98 e Windows 2000 per fornire dati ed eventi relativi ai driver.
Le estensioni WMI per WDM presentano una struttura flessibile, i produttori dei sistemi hardware originali e fornitori di hardware indipendenti (IHV) possono quindi estendere il set di dati fornito di strumenti in base alle proprie esigenze.

REALIZZAZIONE DELL'APPLICAZIONE

Fatta questa breve introduzione su cosa sia e a cosa serva il BIOS e sul WMI, realizziamo la nostra applicazione. Apriamo l'ambiente di sviluppo di Visual Basic e alla richiesta di scelta di un nuovo progetto clicchiamo sulla voce *EXE standard*. Sul form appena creato, andiamo ad inserire 3 oggetti di tipo ListBox che rinomineremo rispettivamente *ListaBios*, *ListaInfo* e *ListaCaratteristiche* e due oggetti CommandButton che chiameremo *CheckBios* ed *Esci*. Diamo uno sguardo ora al codice cominciando dalle dichiarazioni:

```
Option Explicit
Private SWS As SwbemServices
```

Un oggetto di tipo *SwbemService*, realizza operazioni riguardanti un Namespace di un host locale o remoto. Un *Namespace* non è altro che l'oggetto di più alto livello nel WMI data ed essenzialmente rappresenta l'inizio di un percorso verso un oggetto di tipo WMI.
L'associazione al Namespace che ci interessa, viene eseguita nella *Sub Form_Load:*

```
Private Sub Form_Load()
   Set Namespace = GetObject("winmgmts:")
End Sub
```

La stringa *"winmgmts"* connette il Namespace al percorso *root\cimv2* sul computer locale.
Il Namespace di default, viene determinato tramite il valore contenuto nella chiave di registro contenuta in;

```
KEY_LOCAL_MACHINE\Software\Microsoft\WBEM
                              \Scripting\Default Namespace.
```

Analizziamo come si ricava l'elenco dei BIOS presenti nel nostro sistema hardware:

```
Private Sub ElencoBios()
     Dim Bios As SwbemObject
     Dim BiosSet As SWbemObjectSet
     ListaBios.Clear
     Me.MousePointer = vbHourglass
     Set BiosSet = Namespace.InstancesOf(
                                             "Win32_BIOS")
     For Each Bios In BiosSet
          ListaBios.AddItem Bios.Path_.RelPath
     Next
     Set Bios = Nothing
     Set BiosSet = Nothing
     Me.MousePointer = vbNormal
End Sub
```

L'oggetto di tipo *SwbemObject* rappresenta una classe di definizione per la *Windows Management Instrumentation*. *SwbemObjectSet* non è altro che una collezione di oggetti di tipo *SwbemObject*. Tramite il ciclo di *For Each*, ricaviamo tutti i BIOS della collezione e li inseriamo nella *ListBox* che avevamo precedentemente chiamata *ListaBios*. In conclusione, come di consueto, distruggiamo gli oggetti che ormai non ci occorrono più, settandoli a *Nothing*. Ricavato l'elenco dei BIOS presenti nel nostro hardware, ed inserito nell'apposita lista, desideriamo che alla selezione della voce che ci interessa, vengano visualizzate nelle apposite ListBox, tutte le informazioni e le caratteristiche.
Attraverso l'istruzione:

```
Set Bios = Namespace.Get(BIOS selezionato)
```

ricaviamo tutte le informazioni inerenti il BIOS selezionato ed in particolare, *Numero di build, Nome del BIOS, Code Set, Linguaggio corrente, Descrizione, Codice identificativo, Data di installazione, Linguaggio dell'edizione, Produttore, Data ultimo aggiornamento, Numero di serie, Versione BIOS* e *Stato*. Inseriamo tutte queste informazioni nella ListBox *ListaInfo*:

```
Private Sub CaratteristicheBios()
...
   ListaInfo.Clear
   Me.MousePointer = vbHourglass
   SelectedItem = ListaBios.List(ListaBios.ListIndex)
   Set Bios = Namespace.Get(SelectedItem)
   With ListaInfo
      Value = Bios.BuildNumber
       .AddItem "Numero di build: " & CStr(Value)
       Value = Bios.Caption
       .AddItem "Nome: " & CStr(Value)
       Value = Bios.CodeSet
       .AddItem "Code Set: " & CStr(Value)
       Value = Bios.CurrentLanguage
       .AddItem "Linguaggio corrente: " & CStr(Value)
       Value = Bios.Description
       .AddItem "Descrizione: " & CStr(Value)
       Value = Bios.IdentificationCode
```

Sistema

BIOSInfo
Recuperare le informazioni da VB

BIOS

Il BIOS si incarica di eseguire una procedura di autodiagnostica (POST, Power On Self Test) che precede a tutta una serie di controlli e verifiche sulla memoria RAM, tastiera, processore, drive, disco fisso, porte di comunicazione e che permette, infine, di caricare il sistema operativo da floppy disk o da hard disk.

Sistema

BIOSInfo
Recuperare le informazioni da VB

```
.AddItem "Codice identificativo: " & CStr(Value)
Value = Bios.InstallDate
.AddItem "Data di installazione: " & CStr(Value)
Value = Bios.LanguageEdition
.AddItem "Linguaggio dell'edizione: "
                                        & CStr(Value)
Value = Bios.Manufacturer
.AddItem "Produttore: " & CStr(Value)
Value = Bios.Name
.AddItem "Nome: " & CStr(Value)
Value = Bios.PrimaryBIOS
.AddItem "BIOS primario: " & CStr(Value)
Value = Bios.ReleaseDate
.AddItem "Data ultimo aggiornamento: "
                                        & CStr(Value)
Value = Bios.SerialNumber
.AddItem "Numero di serie: " & CStr(Value)
Value = Bios.SMBIOSBIOSVersion
.AddItem "Versione BIOS: " & CStr(Value)
Value = Bios.SoftwareElementID
.AddItem "Software Element ID: " & CStr(Value)
Value = Bios.Status
.AddItem "Stato: " & CStr(Value)
Value = Bios.Version
.AddItem "Versione: " & CStr(Value)
End With
…
```

Fatto ciò, ricaviamo anche le caratteristiche:

```
…
With ListaCaratteristiche
  Value = Bios.BiosCharacteristics
  For tmpInt = 0 To 49
    Select Case Value(tmpInt)
      Case 0: .AddItem "Riservato"
      Case 1: .AddItem "Riservato"
      Case 2: .AddItem "Sconosciuto"
      Case 3: .AddItem "Caratteristiche del BIOS
                                 non supportate"
      Case 4: .AddItem "ISA supportato"
      Case 5: .AddItem "MCA supportato"
      Case 6: .AddItem "EISA supportato"
      Case 7: .AddItem "PCI supportato"
      Case 8: .AddItem "PC Card (PCMCIA)
                                    supportato"
      Case 9: .AddItem "Plug and Play supportato"
      Case 10: .AddItem "APM supportato"
      Case 11: .AddItem "BIOS aggiornabile"
      Case 12: .AddItem "BIOS shadowing consentito"
      Case 13: .AddItem "VL-VESA supportato"
      Case 14: .AddItem "Supporto ESCD disponibile"
      Case 15: .AddItem "Boot da CD supportato"
      Case 16: .AddItem "Avvio selettivo supportato"
      Case 17: .AddItem "BIOS ROM is socketed"
      Case 18: .AddItem "Boot da PC Card (PCMCIA)
                                    supportato"
      Case 19: .AddItem "Specifiche EDD (Enhanced
                          Disk Drive) supportate"
      Case 20: .AddItem "Int 13h - Japanese Floppy
        for NEC 9800 1.2mb (3.5, 1k Bytes/Sector,
                          360 RPM) supportato"
      Case 21: .AddItem "Int 13h - Japanese Floppy
        for Toshiba 1.2mb (3.5, 360 RPM) supportato"
      Case 22: .AddItem "Int 13h - 5.25 / 360 KB
                    Floppy Services supportati"
      Case 23: .AddItem "Int 13h - 5.25 /1.2MB
                    Floppy Services supportati"
      Case 24: .AddItem "Int 13h - 3.5 / 720 KB
                    Floppy Services supportati"
      Case 25: .AddItem "Int 13h - 3.5 / 2.88 MB
                    Floppy Services supportati"
      Case 26: .AddItem "Int 5h, Print Screen
                           Service supportato"
      Case 27: .AddItem "Int 9h, 8042 Keyboard
                          services supportato"
      Case 28: .AddItem "Int 14h, Serial Services
                                   supportato"
      Case 29: .AddItem "Int 17h, printer services
                                   supportato"
      Case 30: .AddItem "Int 10h, CGA/Mono Video
                          Services supportati"
      Case 31: .AddItem "NEC PC-98"
      Case 32: .AddItem "ACPI supportato"
      Case 33: .AddItem "USB supportato"
      Case 34: .AddItem "AGP supportato"
      Case 35: .AddItem "Boot da I2O supportato"
      Case 36: .AddItem "Boot da LS-120 supportato"
      Case 37: .AddItem "Boot da ATAPI ZIP Drive
                                   supportato"
      Case 38: .AddItem "Boot da Firewire supportato"
      Case 39: .AddItem "Smart Battery supportata"
      Case Else: Exit For
    End Select
  Next tmpInt
  End With
  Set Bios = Nothing
  Me.MousePointer = vbNormal
End Sub `CaratteristicheBios
```

La parte clou del nostro programma è praticamente terminata. Non ci resta che associare il codice ai *CommandButton*.
Banalmente avremo:

```
Private Sub CheckBios_Click()
  ElencoBios
End Sub
Private Sub Esci_Click()
  End
End Sub
```

CONCLUSIONI

Non ci resta, ormai, che provare l'applicativo. Per eventuali chiarimenti, è possibile visualizzare il codice sorgente fornito a corredo dell'articolo.

Raffaele Verre

WMI

WMI è un'infrastruttura standardizzata per il monitoraggio e la gestione delle risorse del sistema. WMI consente agli amministratori di sistema di monitorare e controllare il sistema tramite scripting e applicazioni di terze parti.

Stampe

E CREAZIONE DI FILE PDF IN JAVA

In quest'articolo descriveremo un metodo semplice e immediato per effettuare stampe in Java attraverso la creazione di file PDF. La tecnica proposta può essere una valida alternativa a XML/XSLT.

Sistema

File sul CD
\soft\codice\PDF_java.zip

Chi nella sua breve o lunga vita di programmatore, non si è trovato a dover implementare una piccola funzione di stampa, anche solo per stampare la lista della spesa per la mamma? Tale problematica (come del resto tutte), ha mille soluzioni, alcune più complesse altre meno e tutte discutibilmente valide; anche se non possiamo negare che la soluzione più veloce e meno onerosa risulta sempre, ai nostri occhi, come la migliore! Bene quello che voglio proporvi in questo articolo è proprio una soluzione veloce, poco onerosa e soprattutto che funzioni per creare file *.pdf*, da applicazioni Java, pronti per essere stampati. Se dovessimo seguire le mode del momento, quando parliamo di implementare una funzione di stampa, la prima cosa che ci verrebbe in mente è sicuramente: "Bene utilizziamo XML e XSL/XSLT". Tecnologie sicuramente validissime nonché molto alla moda; è innegabile che tutte queste X… sono come il prezzemolo: ormai le trovi ovunque.

QUANDO E PERCHÉ NON UTILIZZARE XML/XSL

Prima di addentrarci nelle situazioni in cui è meglio non utilizzare XSL è doveroso spendere due parole su XSL-FO. XSL-FO ovvero *Extensible Stylesheet Language(XSL) - Formatting Objects(FO)* è un sottoprotocollo XML che consente di descrivere in generale il layout di una pagina e, nello specifico, una stampa. Uno *stylesheet* (foglio di stile) utilizza il linguaggio di trasformazione XSL per trasformare un documento XML con una dato vocabolario (semantico) in un nuovo documento XML (sempre XML) che utilizza il vocabolario XSL-FO (dalla presentazione si passa ad un insieme di tag XML/XSL che hanno una semantica di presentazione). Ricapitolando, l'obiettivo di XSL-FO è quello di applicare uno stylesheet ad un documento, proprio come si fa in HTML. Attualmente i WEB Browser sono capaci di riconoscere gli styleshett CSS *(Cascading Style Sheets)* e fare in modo che venga visualizzato il documento finale. Tale operazione di "riconoscimento" non è ancora possibile per i documenti XML contenenti formattazioni XSL, per questo motivo è necessario compiere un ulteriore passo prima di presentare i documenti in output; questo passo è la trasformazione XSL che viene eseguita mediante XSL-FO. È importante ricordare che il documento di base di tipo XML ha un vocabolario che descrive la semantica dei dati in esso contenuti, con la trasformazione XSL, diventa un nuovo documento sempre XML ma contentente un vocabolario di tag della presentazione, ma rimane comunque un documento XML. Utilizzare XML e affini per implementare una funzione di stampa ha sicuramente i suoi vantaggi:

- XML/XSL è una tecnologia standard.
- XML/XSL è supportata da quasi tutti linguaggi di programmazioni, nelle piattaforme J2EE e .NET.
- In XML/XSL c'è una netta separazione tra il modello dei dati e la presentation.
- In XML/XSL creare un XSL-FO per la formattazione del testo in uscita non implica ricompilare il codice dell'applicazione, anche se si fanno modifiche nella rappresentazione.
- La trasformazione del testo può avvenire in diversi formati di output come Testo, HTML, PDF o PostScript, immagini GIF o altro.
- È possibile creare uno standard di comunicazione e quindi fare il porting dei documenti su differenti piattaforme.

Ma presenta anche alcuni svantaggi:

- Definire dei layout complessi non è un'operazione banale: è necessario un attento studio di XSL-FO per ottenere discreti risultati.
- Si perde un po' di incapsulamento, in quanto i file XSL-FO sono dei normali file di testo che possono essere modificati da chiunque conosca il linguaggio. Questo significa che qualsiasi persona/cliente che utilizza XML-FO per le stampe può modificarle a suo piacimento, e questo non è buono, specie se si vuole guadagnare soldi sull'assistenza!
- Nelle stampe di grandi dimensioni, va preso in considerazione il problema: "Collo di bottiglia".

Il formato PDF

Il Portable Document Format è stato progettato per l'interscambio di documenti in modo che l'utente visualizzi il documento esattamente come è stato creato, indipendentemente dalle diverse elaborazioni fatte su di esso. La principale differenza con gli altri formati è che il PDF è concepito come un formato di descrizione delle pagine; ogni pagina contiene le informazioni utili alla visualizzazione e può essere isolata dalle altre. Il PDF viene generato dal PostScript ma non ne contiene la complessità, per cui una pagina generata da diversi programmi sarà sempre la stessa. Il formato è progettato con buone caratteristiche di sicurezza e crittografia, e la sua diffusione in Rete ha fatto nascere numerosi software freeware che permettono la visualizzazione di tale tipo di documenti.

Sistema

Stampe
e creazione di file PDF in Java

- La trasformazione XSLT è a volte un'operazione "pesante" e, se usata male, potrebbe creare grossi problemi di prestazioni.

Indipendentemente da questi vantaggi/svantaggi, la strada XML/XSLT è sicuramente quella più flessibile, generale e standard. Ma a volte, in semplici sistemi, per effettuare delle stampe possono essere utilizzate metodologie più veloci e dirette. Insomma, per catturare una farfalla basta il retino! Quello che vi propongo è di stampare, o meglio creare dei file *.pdf* utilizzando una semplice libreria free e open-source chiamata *iText*.

LA LIBRERIA ITEXT

Le classi iText risultano molto utili per chi ha necessità di generare documenti con le seguenti caratteristiche:

- Indipendenti dalla piattaforma.
- Contenenti testi, liste, tabelle e immagini.

La libreria in oggetto, presenta numerosi vantaggi se utilizzata in combinazione con Java e tecnologie basate su servlet, poiché mentre il "look and feel" HTML è dipendente dal browser, con queste librerie abbiamo la possibilità di controllare esattamente il "look and feel" di output dalle nostre servlet. Le librerie in questione può essere scaricata gratuitamente all'indirizzo *http://www.lowagie.com/iText* e può essere utilizzata col JDK 1.2 (o versioni successive) e Adobe Acrobate Reader. La libreria iText è presente anche nel CD allegato alla rivista. Diamo adesso uno sguardo alle classi che verranno utilizzate nell'esempio:

- L'oggetto *com.lowagie.text.Document* rappresenta un documento generico cui possono essere aggiunti tutti i tipi di elementi di testo. Possiede metodi che ci consentono di specificare diversi attributi: *Footer, Header, PageSize, Alignment, PageCount, rightMargin, leftMargin* e tanti altri.

- L'oggetto *com.lowagie.text.Table*, è rappresentato come un rettangolo contenente tante celle ordinate come una matrice. È un oggetto fondamentale per la costruzione del layout di stampa e quindi per la visualizzazione dei dati. Anche per questo oggetto esistono diversi metodi che consentono di sfruttare al massimo la sua potenza con il minimo sforzo, con risultati grafici e strutturali degni di nota! La gestione dell'oggetto è pressoché identica alla gestione delle tabelle in HTML, quindi avremmo tutta una serie di metodi che consentono di settare i principali attributi: *Spacing, Cellpadding, Width, Alignment, SpaceBetweenCells, ecc..*

- L'oggetto *com.lowagie.text.Chunk*, rappresenta la più piccola parte di testo che può essere aggiunta ad un Document. Può rappresentare una parte di un elemento da aggiungere, può avere delle caratteristiche proprie come ad esempio font, size ecc..

- L'oggetto *com.lowagie.text.Phrase*, può essere composta da una serie di Chunk. Esso presenta delle caratteristiche proprie (font, size,...) ed eredita quelle del Chunk impostate precedentemente.

- L'oggetto *com.lowagie.text.Paragraph*, può essere composto da una serie di Chunk o Phrase; anche questo mantiene le stesse regole di Phrase. Ha caratteristiche proprie, ma ogni oggetto che lo compone può mantenere le proprie (impostate precedentemente).

- L'oggetto *com.lowagie.text.pdf.PdfWriter*, è fondamentale per la scrittura dei file *.pdf*. Quando un oggetto di tipo *PdfWriter* viene aggiunto ad un Document, ogni elemento aggiunto a questo verrà scritto nello stream di output.

Le classi descritte, rappresentano solo una piccola anzi piccolissima parte delle classi presenti nella libreria *iText*, ma vi assicuro che sono sufficienti per procedere con la nostra prima stampa.

UN ESEMPIO PRATICO

Ok, dopo tante parole vediamo un esempio pratico di utilizzo di queste classi. Ipotizziamo di voler stampare una fattura per una profumeria. I passi necessari saranno i seguenti:

- Importiamo e compiliamo nel nostro ambiente di sviluppo la libreria *iText*. Se usate il compilatore a riga di comando assicuratevi che il jar sia presente nella variabile d'ambiente *CLASSPATH*.

- Importiamo e compiliamo nell'ambiente di svilup-

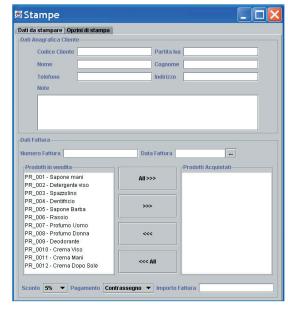



Fig. 1: **Il tab folder "Dati da stampare" della form di stampa.**

La libreria iTEXT

Creata nel 1999 da Bruno Lowagie, ingegnere di Gent, nel Belgio, iText è stata migliorata ed ampliata nel suo complesso anche grazie all'apporto successivo dell'ing. Paulo Soares, attuale corresponsabile del progetto, ed è in continua evoluzione. *iText* è una libreria gratuita che permette di generare file PDF in tempo reale, ossia senza l'uso di file temporanei. Le classi generate mediante *iText* si utilizzano quando si ha necessità di creare file PDF in cui occorre inserire testo, immagini e tabelle, senza utilizzare un lettore di tale formato, ossia senza lavorare sul file da modificare, ma facendo agire il programma Java; tale libreria è usata in combinazione alle tecnologie basate su Java e realizzate mediante l'uso di questo linguaggio di programmazione; richiede almeno la versione 1.2 del *Java Development Kit* (JDK) di Sun.

Fig. 2: Il tab folder "Opzioni di stampa" della form dei stampa.

po il progetto *StampeJava*, e disponibile sul CD allegato alla rivista.

- Eseguiamo il *RunMain* della classe *Run*, ciò che comparirà è la dialog di Fig. 1 e Fig. 2.
- A questo punto non vi rimane che inserire i dati richiesti e premere il pulsante *Stampa…* La fattura è pronta! Nella directory in cui viene eseguito il programma troverete la vostra fattura in formato *pdf*.

Risulta interessante analizzare ciò che concerne la generazione del file *pdf*. Il progetto *StampeJava* è composto da un certo insieme di classi; quelle che richiedono la nostra attenzione sono sostanzialmente due: *GeneraPdf* e *StampePanel*. Vediamo cosa succede quando l'utente preme il pulsante *stampa…*

```
public class StampePanel
{ private JButton _btnStampa = new Jbutton();
public void actionPerformed(ActionEvent ev)
  {  if(ev.getSource().equals(_btnStampa))
      {_generaPdf.stampa(get());} }
}
```

Come si puo notare il metodo *actionPerformed* richiama il metodo stampa appartenente alla classe *GeneraPdf*. Tale metodo si occupa della creazione "fisica" del file *.pdf* e come unico parametro prevede una struttura dati di tipo *StrutturaFattura*, caricata con i dati inseriti dall'utente nella form d'inserimento.

```
public StrutturaFattura get()
{  StrutturaFattura struttura = new StrutturaFattura();
   struttura.codiceCliente = _txtCodiceCliente.getText();
   struttura.partitaIva = _txtPartitaIva.getText();
   struttura.nome = _txtNome.getText();
   struttura.cognome = _txtCognome.getText();
   struttura.telefono = _txtTelefono.getText();
   struttura.indirizzo = _txtIndirizzo.getText();
   struttura.note = _txtNote.getText();
   struttura.numeroFattura = _txtNumeroFattura.getText();
   struttura.dataFattura = _calendario.getStringValue();
   struttura.prodotti = _doubleList.getAssociatedData();
   return struttura; }
```

Il metodo *get* della classe *StampaPanel* ha il compito di recuperare i dati inseriti dall'utente attraveso la form d'inserimento, e di riempire con essi una struttura dati di tipo *StrutturaFattura*.

```
public class GeneraPdf
{ private com.lowagie.text.Document _document = null;
PdfWriter _writer = null;
   private StrutturaFattura _strutturaFattura = null;
private String _font = "";
   private String _orientamento = "";
private String _color = ""; }
```

La classe *GeneraPdf* rappresenta il motore dell'applicazione. Sostanzialmente svolge diverse operazioni:

- Crea fisicamente il file *.pdf*.
- Crea il layout di stampa, ogni singola riga contenuta nella stampa viene creata all'interno di questa classe.
- Scrive sul "foglio bianco" della nuova stampa i dati inseriti dall'utente per la creazione di una nuova fattura.

Di seguito vengono analizzati alcuni metodi appartenenti a questa classe. Il primo è il seguente:

```
public boolean stampa(StrutturaFattuta strutturaFattuta)
{ _strutturaFattura = strutturaFattura;
   _font = _strutturaFattura.font;
   _color = _strutturaFattura.colore;
   _orientamento = _strutturaFattura.orientamento;
   _fontSize= new Integre(_strutturaFattura.fontSize)..
   String nomeFile = null;
nomeFile = _strutturaFattura.numeroFattura + ".pdf";
_document = new com.lowagie.text.Document();
_document.addTitle("Fattura");
_writer = PdfWriter.getInstance(_document,
new FileOutputStream(nomeFile));
if(_orientamento.trim().equalsIgnoreCase(
                      Costanti.ORIENTAMENTO_ORIZZONTALE))
_document.setPageSize(PageSize.A4.rotate());
_document.open();
generaLayoutDatiAnagraficiCliente();
_document.close(); }
```

Si tratta del metodo stampa sopra descritto; viene richiamato dalla classe *StampaPanel* alla pressione del pulsante *"stampa"*. Appartiene alla classe *GeneraPdf*. Il suo compito è quello di settare nella nuova stampa i parametri: *colore*, *font*, *orientamento*, *fontSize*, *StrutturaDati* (di tipo *StrutturaFattura*) impostati dall'utente in fase di creazione fattura ed inizializzare gli oggetti più importanti, come *Document* e *PdfWriter*, fondamentali per la creazione di una nuova stampa; infine richiama il metodo *generaLayoutDatiAnagraficiCliente*, che si occupa delle creazione del Layout contenente i dati anagrafici del cliente inseriti in fattura.

Sistema

Stampe
e creazione di file PDF in Java

Protezione dei documenti

iText permette la protezione dei documenti con due livelli di crittazione del documento, una a 40bit e l'altra a 128bit. La libreria permette diversi livelli di accesso al documento da parte dell'utente, mediante l'utilizzo di alcuni metodi predefiniti:

- ***AllowPrinting:*** **permette la stampa del documento**
- ***AllowModifyContents:*** **consente la modifica dei contenuti del file**
- ***AllowCopy:*** **autorizza il copia-incolla**
- ***AllowModifyAnnotations:*** **concede la modifica delle note**
- ***AllowFillIn:*** **consente la compilazione di eventuali parti del documento, ad esempio una form**
- ***AllowScreenReaders:*** **permette la sola visualizzazione del file**
- ***AllowAssembly:*** **autorizza l'inclusione di oggetti nel documento, come immagini o grafici;**
- ***AllowDegradedPrinting:*** **concede la stampa degradata, in modo da non renderla perfetta;**

Tutti questi metodi di crittazione possono essere inseriti nel documento con o senza password, inoltre si può decidere se dare una password proprietario ed una utente, consentendo quindi una modifica successiva da parte del creatore del file.

Sistema

Stampe
e creazione di file PDF in Java

I tre passi per creare un PDF con iTEXT

Per realizzare dei file PDF mediante iText, si devono seguire passi piuttosto semplici, che possono essere divisi in tre fasi:

- **Creazione del documento vuoto, che conterrà il file: in questa fase è possibile fissare i margini, le dimensioni della pagina, la sua rotazione e il nome del file PDF risultante.**
- **Apertura documento ed elaborazione: in questa fase avviene la modifica vera e propria del PDF; si possono importare pagine, aggiungere del testo, sia come frasi, se si tratta di un file vuoto, sia come box di testo da posizionare sulla pagina, aggiungere l'intestazione e il piè pagina, immagini, ecc...**
- **Chiusura del documento: normalmente questa fase è compiuta in automatico dal programma Java, ma è consigliabile sempre chiudere il documento, una volta terminate le modifiche, all'interno del programma, poiché si ha la sicurezza che le risorse impegnate dal programma verranno rilasciate e il documento sarà utilizzabile.**

Il metodo seguente:

```
private boolean generaLayoutDatiAnagraficiCliente()
{ Table aTable = new Table(2,8);
  aTable.setAutoFillEmptyCells(true);
  aTable.setWidth(105);
  aTable.setBorderWidth(0);
  aTable.setPadding(1);
  Phrase p = new Phrase(CODICE_CLIENTE + ": ",
      FontFactory.getFont(FontFactory.HELVETICA, 12));
  Cell cellCodiceCliente = new Cell(p);
  aTable.addCell(cellCodiceCliente, new Point(1,0));
  String valoreCodiceCliente = _strutturaFattura.codiceCliente;
  Cell cellCodiceClienteVal = new Cell(getString(
                                    valoreCodiceCliente));
  aTable.addCell(cellCodiceClienteVal, new Point(1,1));
  _document.add(aTable);  }
```

viene richiamato all'interno del metodo *stampa* che, come detto in precedenza, rappresenta uno dei metodi fondamentali della classe *GeneraPdf*. Si occupa di generare il layout della stampa relativo ai dati anagrafici del cliente e di aggiungere i dati del cliente contenuti in *_strutturaFattura* al *Document* (*oggetto _document*). Per la creazione di un layout è fondamentale comprendere ogni operazione fatta all'interno di questo metodo, quindi lo analizzeremo passo passo. Attraverso l'istruzione *Table aTable = new Table(2, 8)*; creaiamo un nuovo oggetto di tipo *Table* e specifichiamo che la tabella deve avere 2 colonne e 8 righe.
Come si può notare, per il nuovo oggetto *Table*, impostiamo una serie di attributi che ne caratterizzano il layout. Tali attributi sono pressoché identici ai Tag utilizzati nelle tabelle HTML.
L'istruzione:

```
Phrase ph = new Phrase(CODICE_CLIENTE + ": ",
FontFactory.getFont(FontFactory.HELVETICA, 14)))
```

crea un nuovo oggetto di tipo *Phrase*. Questo rappresenta una semplice "frase" alla quale può essere associata una qualsiasi stringa. Anche per l'oggetto in questione abbiamo la possibilità di specificare diversi attributi: *color, font, fontSize, underline*, ecc.. Nel nostro caso, l'oggetto ph contiene la Stringa *"Codice Cliente :"*, il suo font è *HELVETICA*, e il *fontSize 14*.
Andando avanti incontriamo l'istruzione

```
Cell cellCodiceCliente = new Cell(ph);
```

che crea un nuovo oggetto *Cell*. Nel costruttore passiamo l'oggetto *Phrase* creato nello step precedente. Il risultato sarà quello di visualizzare la stringa *"Codice Cliente"* all'interno della cella creata. Anche per questo oggetto abbiamo la possibilità di impostare numerosi attributi, e ancora una volta molti di questi sono identici ai tag definiti in una cell HTML. Con l'istruzione:

```
aTable.addCell(cellCodiceCliente, new Point(0,0));
```

Dati Anagrafica Cliente:

Codice Cliente:	1
Partita Iva:	MZLAFDRC12488
Nome:	Federico
Cognome:	Mazzola
Telefono:	02/889967
Indirizzo:	Via Roma 10
Note: Consegnare la merce dopo le 18.	

Fig. 3: **Il risultato di una stampa.**

aggiungiamo la nuova *Cell* (oggetto *cellAnagraficaCliente*) alla *Table* (oggetto *aTable*) creato precedentemente. Tale operazione è realizzata mediante il metodo *addCell*, che prevede nel costruttore la cella da aggiungere e la posizione precisa nella quale la cella deve essere collocata all'interno della *Table*. Infine con

```
_document.add(aTable);
```

la *Table* è aggunta al Document. Quest'ultima operazione è estremamente importante se non vogliamo che il risultato di tutti sia un bellissimo foglio bianco! Il risultato, del codice completo disponibile con la rivista, è mostrato in Fig. 3. Un altro metodo importante da analizzare è getString.
Il codice è il seguente:

```
private Phrase getString(String valore)
{ Phrase phrase = null;
  if(_font.equalsIgnoreCase(COURIER))
  {  phrase = new Phrase(valore,
      FontFactory.getFont(FontFactory.COURIER,
                            _fontSize, 2,getColor())) ;}
  Return phrase; }
```

Si tratta di un metodo privato della classe *GeneraPdf* che, data una stringa in input, restituisce un oggetto di tipo *Phrase* contenente la stringa stessa modificata secondo i parametri *(colore, font, fontSize)* inseriti dall'utente. Di questo metodo ho omesso buona parte del codice sorgente, poiché non è fondamentale per la comprensione della costruzione del Layout di stampa. Lo stesso discorso vale per gli esempi proposti. Pertanto sarà il caso di compilare il codice sorgente incluso ne CD-Rom allegato, all'interno dell'ambiente di sviluppo, per ottenere gli stessi risultati dell'articolo.

CONCLUSIONI

Siamo arrivati alle conclusioni, spero che in questo momento almeno la metà di voi sia già alle prese con le potenti classi *iText*. Mi rendo conto che in un primo momento possono sembrare poco interessanti nonché poco funzionali, ma vi assicuro che la loro potenza nonché la loro facilità d'utilizzo è grandiosa. L'unica cosa che posso dirvi è di dedicargli 30 minuti del vostro tempo: implementate 2 classi e provatele… Se avete bisogno di me mi trovate all'indirizzo e-mail: *valentina.muraglia@tiscali.it*.

Valentina Muraglia

JSP

JavaBean

E PAGINE JSP

Il modello di programmazione offerto dai componenti JavaBean, è uno tra i tanti strumenti a disposizione dello sviluppatore JSP per il raggiungimento dello scopo.

A livello elementare, un JavaBean è una classe Java che soddisfa i seguenti requisiti:

- Ha un costruttore privo di argomenti.
- Implementa l'interfaccia *java.io.Serializable*.

Come è possibile osservare, si tratta di una definizione assai generica. Molte classi rientrano, infatti, nella categoria dei JavaBean, ed altrettante possono facilmente essere adattate ai requisiti richiesti. In secondo luogo, lo scopo di un JavaBean è offrire delle proprietà che possano essere sfruttate in lettura o in scrittura dal codice chiamante, secondo delle precise ma semplici norme d'impiego. Le specifiche di JavaBean prevedono un modello di scrittura del codice che rispecchi quanto mostrato nel seguente esempio:

```
import java.io.*;
public class MrJavaBean implements Serializable {
  private String nome;
  private String cognome;
    public void setNome(String n) {
    nome = n; }
    public String getNome() {
      return nome; }
    public void setCognome(String c) {
      cognome = c; }
    public String getCognome() {
        return cognome; }
    public String getFirma() {
        return nome + " " + cognome; }
}
```

La classe *MrJavaBean* offre funzionalità per la rappresentazione di una persona fisica, attraverso le sue proprietà *nome* e *cognome*. *MrJavaBean* soddisfa ambo i requisiti richiesti:

- Non ha alcun costruttore esplicitamente dichiarato. Questo significa che la classe sarà implicitamente dotata di un solo costruttore privo di argomenti, come richiesto dalla definizione mostrata sopra.
- Implementa l'interfaccia *java.io.Serializable*. Questa speciale interfaccia non richiede l'implementazione di metodi specifici. Semplicemente, funziona da marcatore: una classe che implementa *Serializable* può essere serializzata. Tutto qui! Sui significati e gli scopi della serializzazione, per chi non è avvezzo a questo aspetto del linguaggio Java, torneremo tra poco.

Inoltre, la stesura di *MrJavaBean* è stata svolta osservando un particolare criterio:

- Le proprietà caratteristiche della classe, vale a dire le stringhe *nome* e *cognome*, sono ad accesso privato. Non sono esplicitamente esposte al codice esterno alla classe.
- L'accesso, in lettura e scrittura, alle proprietà caratteristiche della classe avviene mediante alcuni metodi pubblici specializzati, tutti nella forma *getNomeProprietà()* e *setNomeProprietà()*.

Questo modello di progettazione permette un controllo dettagliato sui valori delle proprietà caratteristiche della classe. Ad esempio, è possibile restringere il dominio di una proprietà, inserendo appositi controlli nel corrispondente metodo di tipo *set*. Si potrebbero rifiutare alcuni range di valori, come la stringa vuota e *null* nel caso delle proprietà *nome* e *cognome*. Un controllo così dettagliato non sarebbe possibile esponendo pubblicamente le proprietà, senza impiegare dei metodi pronti a lavorare da "filtro". Non solo: con del codice apposito, è possibile adottare un metodo di programmazione basato sugli eventi. Il concetto è lo stesso impiegato dalle classi di *AWT*. Quando una proprietà subisce una modifica, il metodo *set* corrispondente può farne notifica ad un qualsiasi gestore di eventi collegato al JavaBean. *MrJavaBean*, inoltre, è stata dotata del metodo *getFirma()*, che esegue una elementare computazione con i dati accumulati nelle sue proprietà, restituendo il risultato al codice chiamante.

File sul CD
\soft\codice \codici_jsp14.zip

Sul Web
Diversi approfondimenti legati alla lezione odierna e alle precedenti sono reperibili in Internet, tra le dispense Web del corso. L'URL di riferimento è http://www.sauronsoftware.it /dispenseweb/jsp/

JSP

Riassumendo, una classe JavaBean comunica con il mondo esterno principalmente attraverso dei metodi nella forma *get* e *set*. Il funzionamento interno, a questo punto, non è rilevante per chi indossa i panni dell'utilizzatore del bean. Le possibilità di scomporre un software in moduli indipendenti sono notevoli. In linea di massima, questa è la filosofia che permea la realizzazione e l'utilizzo delle classi JavaBean.

LA SERIALIZZAZIONE

La serializzazione è uno dei tasselli fondamentali della tecnologia JavaBean. Tutti i JavaBean sono serializzabili. Nel dialetto di Java, un oggetto è serializzabile quando può essere convogliato in uno stream di dati, per essere conservato sul disco o per essere trasmesso attraverso una rete. Quando lo stream sarà recuperato ed interpretato in senso inverso (operazione detta di deserializzazione), sarà possibile ricostruire l'oggetto, ripristinando l'esatto stato interno che aveva al momento della sua serializzazione. Perno della serializzazione sono i due stream *java.io.ObjectOutputStream* e *java.io.ObjectInputStream*, rispettivamente per la serializzazione e per la deserializzazione degli oggetti.
Il migliore approccio alla serializzazione che può essere offerto, in casi come questo, passa sempre attraverso la dimostrazione di un esempio pratico. Impiegando la classe *MrJavaBean* realizzata in precedenza, tentiamo la sua serializzazione con il seguente codice:

```
import java.io.*;
public class Serializza {
    public static void main(String[] args) {
    // Creazione ed impostazione del bean.
    MrJavaBean mrJavaBean = new MrJavaBean();
    mrJavaBean.setNome("Mario");
    mrJavaBean.setCognome("Rossi");
    // Serializzazione del bean su un file.
    try {
        FileOutputStream fileOut = new FileOutputStream(
                                        "mrjavabean.ser");
        ObjectOutputStream objOut = new
                               ObjectOutputStream(fileOut);
        objOut.writeObject(mrJavaBean);
        objOut.flush();
        objOut.close();
        fileOut.close();
        } catch (Exception e) {System.out.println(e);}
    }
}
```

L'esecuzione della classe genera il file *mrjavabean.ser*, che contiene l'immagine serializzata dell'oggetto *MrJavaBean* generato in apertura, vale a dire di Mario Rossi. In maniera speculare, il file può essere recuperato e deserializzato, come mostrato di seguito:

```
import java.io.*;
public class Deserializza {
  public static void main(String[] args) {
    MrJavaBean mrJavaBean = null;
    // Deserializzazione del bean.
    try {
      FileInputStream fileIn = new FileInputStream(
                                    "mrjavabean.ser");
        ObjectInputStream objIn = new
                              ObjectInputStream(fileIn);
        mrJavaBean = (MrJavaBean)objIn.readObject();
        objIn.close();
        fileIn.close();
        } catch (Exception e) {System.out.println(e);}
        // Verifica del contenuto del bean.
        if (mrJavaBean != null) {
          System.out.println(mrJavaBean.getFirma());}
}   }
```

L'esecuzione di questa classe, se tutto va a buon fine, mostra in output la riga:

```
Mario Rossi
```

Lo stato interno dell'oggetto è stato mantenuto, attraverso le operazioni di serializzazione e deserializzazione. Riassumendo, la serializzazione è una tecnica che fornisce agli oggetti Java delle doti di persistenza. Questo è il secondo importante tassello che caratterizza i JavaBean.

JSP E I JAVABEAN

La digressione svolta sinora è stata senz'altro piacevole. Tuttavia, ancora non è stato chiarito il ruolo dei JavaBean all'interno della tecnologia JavaServer Pages. Ci accingiamo ora a colmare questa lacuna, giungendo al nocciolo della lezione. Benché i JavaBean, in quanto comuni classi Java, possano essere normalmente sfruttati in scriptlet, dichiarazioni ed espressioni, JSP fornisce un metodo di utilizzo dei JavaBean di più alto livello. In particolare, esistono tre azioni che riguardano i JavaBean:

- *<jsp:useBean>* per richiedere l'impiego di un JavaBean.
- *<jsp:setProperty>* per impostare le proprietà di un JavaBean.
- *<jsp:getProperty>* per leggere i valori conservati nelle proprietà di un JavaBean.

<JSP:USEBEAN>

L'azione *<jsp:useBean>* recupera, crea o deserializza un JavaBean, associandolo ad una variabile Java che potrà normalmente essere impiegata negli scriptlet del documento JSP corrente. La sintassi dell'azione deve riflettere il seguente modello:

```
<jsp:useBean id="nome" scope="visibilità" specifiche />
```

oppure:

Maggiori informazioni sui JavaBean

Se siete rimasti affascinati dai JavaBean e siete interessati ad approfondire l'argomento, il punto di partenza ideale è:

http://java.sun.com/products/javabeans/

```
<jsp:useBean id="nome" scope="visibilità" specifiche>
  [corpo dell'azione]
</jsp:useBean>
```

L'attributo *id* indica il nome che dovrà essere associato al JavaBean, in modo che sia successivamente possibile farne uso. Il campo rispetta tutte le norme valide per gli identificatori Java. In particolare, è case-sensitive. L'attributo *scope* permette di specificare la visibilità del bean. Sono possibili, come in altre situazioni presenti nell'ambito JSP, quattro distinti valori:

- *page*. Il JavaBean sarà valido per tutta la durata della pagina JSP corrente. Si tratta del campo di visibilità predefinito, nel caso l'attributo *scope* non venga specificato.
- *request*. Il JavaBean sarà valido per tutta la durata della pagina JSP corrente, come nel caso precedente, e sarà valido anche all'interno di tutti gli altri documenti percorsi dalla richiesta. In sostanza, sarà ripreso anche da tutte le pagine raggiunte mediante delle azioni di tipo *<jsp:include>* e *<jsp:forward>* (o chiamate analoghe).
- *session*. Il JavaBean sarà associato alla sessione utente corrente, e pertanto sarà valido durante tutte le richieste avanzate dal medesimo utente.
- *application*. Il JavaBean sarà universalmente valido e condiviso da tutte le pagine JSP dell'applicazione Web corrente.

Più complesso è l'insieme di attributi riassunto, nello schema mostrato in precedenza, dalla dizione "specifiche". Le specifiche del bean sono costituite da una combinazione di tre distinti attributi: *class*, *type* e *beanName*. Le combinazioni valide sono quattro:

- Solo *type*.
- Solo *class*.
- Insieme, *type* e *class*.
- Insieme, *type* e *beanName*.

A seconda del modello scelto, l'azione *<jsp:useBean>* si comporterà in maniera differente:

- Se viene specificato il solo attributo *type*, l'engine ricercherà nell'ambito di visibilità specificato un JavaBean già esistente, che corrisponda all'identificatore e al tipo desiderati. Nel caso non sia possibile localizzare un bean precedentemente istanziato che rispecchi ambo le caratteristiche, sarà propagata un'eccezione. L'eccezione sarà di tipo *InstantiationException* se, nell'ambito voluto, nessun bean corrisponde all'identificatore specificato. Se il bean esiste, ma non può essere riportato al tipo voluto, l'eccezione sarà una *ClassCastException*. Questa combinazione, riassumendo, va impiegata per richiamare bean già esistenti nell'ambito voluto.
- Se viene specificato il solo attributo *class*, l'engine si comporterà come nel caso precedente, ricercando un bean che soddisfi l'ambito di visibilità, l'identificatore e il tipo (la classe) specificati. Nel caso il bean venga localizzato, ma non sia possibile convertirlo nel tipo espresso, sarà propagata una *ClassCastException*. Al contrario, se il bean non esiste, l'engine lo istanzierà per la prima volta. Questa combinazione, riassumendo, serve sia per richiamare bean esistenti, sia per crearne di nuovi nel caso la ricerca fallisca.
- Se vengono specificati gli attributi *type* e *class*, l'engine seguirà i medesimi passi della combinazione precedente. Il tipo restituito, ad ogni modo, corrisponderà sempre a quanto specificato nell'attributo *type*. In caso di creazione di un nuovo bean, la nuova istanza sarà realizzata basandosi su quanto specificato dall'attributo *class*. Questa combinazione, riassumendo, permette di ottenere o creare dei bean di una certa classe, ottendo indietro solo una delle loro possibili interfacce. Ovviamente, deve esistere un rapporto di relazione tra il tipo e la classe, altrimenti si riceverà una *ClassCastException*.
- Se vengono specificati gli attributi *type* e *beanName*, l'engine ricercherà nell'ambito specificato un'istanza che soddisfi i campi id e *type*, come nel primo caso mostrato. Nel caso non sia possibile localizzare l'istanza, l'engine andrà alla ricerca di un bean serializzato, da deserializzare ed importare all'interno dell'ambito specificato. L'attributo *type* indica il tipo associato al bean, mentre *beanName* localizza lo stream necessario per la deserializzazione. Questo attributo va espresso secondo la comune notazione puntata di Java, tenendo presente che il valore sarà convertito in un percorso che conduca alla serializzazione del bean desiderato. Ad esempio, se *beanName* è *it.mioPacchetto.MiaClasse*, l'engine andrà alla ricerca del file */it/mioPacchetto/MiaClasse.ser*. Se la ricerca del file fallisce, l'attributo *beanName* sarà trattato come un attributo *class*, e pertanto si tenterà una nuova istanziazione. Questa combinazione, riassumendo, può portare a tre differenti esiti: la localizzazione di un bean già esistente nell'ambito voluto, la deserializzazione di un bean precedentemente serializzato, oppure la creazione da zero di una nuova istanza del bean.

Il corpo di *<jsp:useBean>* viene valutato ed eseguito solo se l'azione ha portato all'istanziazione di un nuovo bean.
In caso contrario, è ignorato. Tale corpo può contenere scriptlet e azioni del tipo *<jsp:setProperty>*. Lo scopo del blocco è fornire un'inizializzazione preliminare al nuovo bean.

<JSP:SETPROPERTY>

L'azione *<jsp:setProperty>* imposta le proprietà di un JavaBean, in corrispondenza dei metodi del bean espressi nella già discussa forma *setNomeProprietà()*. Esistono diverse metodologie d'uso di questa azione. La più semplice è la seguente:

BeanBuilder

BeanBuilder è uno strumento visuale per la dimostrazione e la verifica dei componenti JavaBean. Potete prelevarlo dalla pagina:

http://java.sun.com/products/javabeans/beanbuilder/index.html

JSP

```
<jsp:setProperty name="idBean" property="proprietà"
                                      value="valore" />
```

L'attributo name specifica l'identificatore del bean coinvolto dall'azione. Una volta localizzato il JavaBean interessato dall'operazione, la sua proprietà identificata dall'attributo property riceverà, se possibile, il valore espresso in *value*.
Il codice seguente:

```
<jsp:useBean id="mioBean" class="MrJavaBean">
  <jsp:setProperty name="mioBean" property="nome"
                                      value="Mario" />
  <jsp:setProperty name="mioBean"
                  property="cognome" value="Rossi" />
</jsp:useBean>
```

corrisponde, a grosse linee, a:

```
MrJavaBean mioBean = new MrJavaBean();
mioBean.setNome("Mario");
mioBean.setCognome("Rossi");
```

Questo è vero solo nel caso in cui, nell'ambito della pagina corrente, non esista già un'istanza di *MrJavaBean* chiamata *mioBean*. In generale, comunque, l'esempio rende bene l'idea di come lavori l'azione *<jsp:setProperty>*. L'identificatore del bean riferito ed il nome della proprietà coinvolta nell'operazione devono sempre essere espressi letteralmente. Per questi attributi, non sono ammesse forme dinamiche generate mediante espressioni. Pertanto, la seguente azione è scorretta:

```
<jsp:setProperty name="idBean" property=
                         "<%= ... %>" ... /> ERRATO!!!
```

Al contrario, il valore può eventualmente essere dettato dinamicamente, in funzione di un'espressione:

```
<jsp:setProperty
  name="idBean"
  property="proprietà"
  value="<%= ... %>"
/> CORRETTO!!!
```

Un'altra forma di utilizzo dell'azione *<jsp:setProperty>* è la seguente:

```
<jsp:setProperty name="idBean" property="proprietà" />
```

Come è possibile osservare, manca all'appello l'attributo *value*. In questo caso, il valore assegnato alla proprietà viene desunto dai parametri della richiesta. Si tratta di una vantaggiosa scorciatoia per far corrispondere alle proprietà di un bean i valori che l'utente ha digitato in un form HTML.
In pratica, l'azione:

```
<jsp:setProperty name="mioBean" property="nome" />
```

corrisponde a:

```
<jsp:setProperty
  name="mioBean"
  property="nome"
  value="<%= request.getParameter("nome") %>"
/>
```

In alcuni casi, il nome di una proprietà potrebbe non corrispondere al nome del campo HTML che si desidera abbinargli. In situazioni di questo tipo, è valida la forma:

```
<jsp:setProperty name="idBean" property="proprietà"
                                  param="parametro" />
```

Alla proprietà specificata sarà assegnato il valore contenuto nel parametro della richiesta identificato dall'attributo *param*. Infine, esiste una forma estremamente breve per associare i parametri ricevuti alle proprietà di un bean:

```
<jsp:setProperty name="idBean" property="*" />
```

L'engine andrà alla ricerca di tutte le omonimie esistenti tra le proprietà del bean ed i parametri della richiesta, e quindi effettuerà tutte le assegnazioni possibili.

<JSP:GETPROPERTY>

Il recupero dei valori conservati nelle proprietà di un JavaBean può essere effettuato servendosi dell'azione *<jsp:getProperty>*.
La forma generica è la seguente:

```
<jsp:getProperty name="idBean" property="proprietà" />
```

L'azione è equivalente a

```
<%= idBean.getProprietà() %>
```

Ad ogni modo, il contenuto degli attributi *name* e *property*, come nel caso di *<jsp:setProperty>*, deve essere espresso letteralmente. Non può essere generato mediante un'espressione. Pertanto, la seguente forma è scorretta:

```
<jsp:getProperty name="idBean" property="<%= ...
                                   %>" /> ERRATO!!!
```

CONCLUSIONI

Nel corso della prossima lezione, per conferire maggior valore a quanto esaminato in queste pagine, prenderemo in considerazione alcuni esempi pratici, che dimostreranno l'utilizzo e l'efficacia della tecnologia JavaBean nell'ambito Web occupato dalle JavaServer Pages.

Carlo Pelliccia

Bibliografia

- **JSP, LA GUIDA COMPLETA** *Phil Hanna* (McGraw-Hill) ISBN 88-386-4207-9 2001
- **IMPARARE JAVASERVER PAGES IN 24 ORE** *Jose Annunziato, Stephanie Fesler Kaminaris* (Tecniche Nuove) ISBN 88-481-1306-0 2001
- **JAVA, LA GUIDA COMPLETA** *Patrick Naughton & Herbert Schildt* (McGraw-Hill) ISBN 88-386-0416-9 1997
- **JAVA 2, TUTTO & OLTRE** *Jamie Jaworski* (Apogeo) ISBN 88-7303-466-7 1999

Lo sviluppo

ORIENTATO AGLI OGGETTI

Visual Basic .NET

Il termine orientato agli oggetti, negli anni passati era considerato un argomento delicato e difficile da comprendere, ma oggi la OOP è alla base della programmazione moderna ed offre enormi vantaggi agli sviluppatori di software.

Nei precedenti articoli ci siamo impadroniti degli strumenti di base di VB.NET, ed abbiamo acquisito le conoscenze sulla manipolazione delle Windows Form e di alcuni controlli. Da questo articolo affronteremo i componenti VB per la programmazione orientata agli oggetti, iniziando con un pizzico di teoria e descrivendo successivamente come creare una classe con proprietà e metodi.

DALLA PROGRAMMAZIONE STRUTTURATA ALLA PROGRAMMAZIONE AD OGGETTI

La tecnica della programmazione strutturata è stata per anni l'unico aiuto per il programmatore che, osservandone i principi, poteva implementare codice più semplice da scrivere e da modificare. Essenzialmente la programmazione strutturata impone la divisione del programma in piccoli sottoprogrammi dove ogni sottoprogramma esegue esattamente una funzione. La programmazione strutturata ha però alcuni limiti tra i quali la propagazione incontrollata dei dati, modificabili da qualsiasi punto del programma.

Il concetto fondamentale della programmazione ad oggetti è la suddivisione del programma in parti isolate, e indipendenti dalle altre, denominate *oggetti*. Gli oggetti sono concettualmente molto simili ai blocchetti dei giochi di costruzione, se programmati per bene potete disporli per costruire qualcosa di più grande della sola somma delle parti. Per accedere ai dati di un oggetto si deve semplicemente dare un comando all'oggetto che li contiene. Non è mai possibile accedere direttamente ai dati ma soltanto ai comandi che consentono l'accesso ad essi. Per modificare il modo in cui il programma accede ai dati di un oggetto si devono cambiare soltanto i comandi all'interno dell'oggetto. In questo modo la OOP permette di ridurre la possibilità di introdurre nuovi errori nel programma quando è necessaria una qualsiasi modifica.

DEFINIZIONI DI CLASSI ED OGGETTI

In letteratura si definisce *oggetto*: un insieme correlato di dati (proprietà) e funzioni (metodi) che agiscono su tali dati. Oggetti diversi con caratteristiche simili sono raggruppabili in una classe. Così tutti i cittadini di un comune sono oggetti della classe *cittadino*, tutti i clienti di un supermercato sono oggetti della classe *cliente*. Anche per le proprietà ed i metodi si può fare una similitudine con il mondo reale. Il cliente Luigi Bonacci ha un nome ed un cognome definiti tramite le sue proprietà, per conoscere le sue abitudini di spesa e quello che spende ogni anno in derrate alimentari, si deve fare riferimento ad una funzione o metodo dell'oggetto *Luigi Bonacci*, che calcola e restituisce la sua spesa annuale. Sempre in letteratura si afferma che un oggetto è un'istanza di una classe. È bene chiarire dal punto di vista della programmazione la differenza tra classe e oggetto. Una classe è una parte di programma (in VB6 un modulo di classe, in VB .Net un pezzo di codice) che contiene il codice necessario alla definizione di proprietà, metodi ed eventi di oggetti che saranno creati in fase di esecuzione. Un oggetto, invece, è un'area di memoria creata in fase di esecuzione. L'utente non utilizzerà mai una classe, ma piuttosto utilizzerà gli oggetti creati dalle classi, ad esempio i dati di un cliente o lo stato civile di un cittadino. Il programmatore, viceversa, si troverà a scrivere il codice per definire una classe.

CARATTERISTICHE E VANTAGGI DELLA OOP

Incapsulamento - L'incapsulamento è una caratteristica fondamentale della OOP, in pratica, i dati sono pro-

Classe

Chi ha già familiarità con le versioni precedenti di Visual Basic assocerà immediatamente il concetto di classe a quello di modulo di classe. In effetti, le classi di VB .NET mantengono tutte le caratteristiche dei moduli di classe, ma ne aggiungono molte altre. Inoltre, da un punto di vista pratico la creazione di una classe non prevede la creazione di un Modulo di classe, ma semplicemente la stesura di un blocco di codice delimitato dalle istruzioni *"Class"* ed "End Class".

prietari dell'oggetto e sono accessibili e modificabili solo attraverso i suoi metodi. Questa semplice proprietà porta ai seguenti vantaggi:

1) **Integrazione dati-funzioni**

- L'accesso e l'elaborazione del dato viene effettuata in un UNICO punto;
- Abbattimento delle ridondanze nel codice.

2) **Protezione dei dati**

- L'accesso ai dati di un oggetto è controllato dall'oggetto stesso, che pertanto ne garantisce l'integrità.

3) **Indipendenza delle modifiche**

- La struttura interna di un oggetto può cambiare senza impatti sugli altri oggetti;
- Minimizzazione dell'impatto sugli altri oggetti nel sistema;
- Facilità di manutenzione.

L'incapsulamento permette di creare classi facilmente riutilizzabili che non dipendono da un'altra entità esterna. Nell'esempio precedente si è fatto riferimento ad una funzione o metodo dell'oggetto *cliente*, che calcola e restituisce la sua spesa annuale in derrate alimentari, in questa funzione è incapsulata la logica in cui si calcola la spesa, in modo tale che chiunque utilizzi la funzione possa accedere al valore della spesa, senza dover sapere come è effettivamente eseguita l'operazione.

POLIMORFISMO

Banalmente polimorfismo significa "molte forme". Applicando questo termine alla OOP, il polimorfismo è la caratteristica per cui classi diverse hanno metodi con lo stesso nome che implementano comportamenti diversi. Un esempio classico è quello di un'applicazione in cui è necessario salvare i dati delle classi *Fattura* e *Cliente* nel database, tutte e due le classi avranno il metodo *Salva*, ma naturalmente esso sarà implementato in maniera differente poiché i dati della classe cliente sono differenti dai dati della classe *Fattura*. Un altro esempio è quello di oggetti completamente diversi tra loro quali: TextBox, ComboBox, Label che espongono alcuni metodi e proprietà uguali. Grazie a questa caratteristica il programmatore, non avrà necessità di ricordare nomi e sintassi diverse. Sfruttando il polimorfismo è possibile, ad esempio, scrivere procedure generiche che agiscono sulle proprietà di tutti i controlli presenti su una form riducendo notevolmente la quantità di codice da scrivere.

EREDITARIETÀ

L'ereditarietà è la capacità di una classe (classe derivata o sottoclasse) di ereditare o ottenere funzionalità di un'altra classe (la classe base o superclasse). In questo modo la classe derivata eredita automaticamente le proprietà ed i metodi della classe. I vantaggi dell'ereditarietà possono essere riassunti in:

1) **Riusabilità**

- Una classe può essere costruita in modo indipendente da uno specifico contesto;
- Tutte le caratteristiche generali sono scritte una sola volta.

2) **Portabilità**

- Cambiando il contesto, è sufficiente cambiare la classe base, senza modifiche sulla parte applicativa.

3) **Riduzione del codice**

- Non è necessario riscrivere tutti i metodi e le proprietà della classe base ma soltanto quelli peculiari della classe derivata.

La parola chiave utilizzata in VB.Net per creare una relazione di eredità è *Inherits*

Overloading - L'overloading è un'altra caratteristica dei linguaggi OO implementata in VB .Net, sotto certi aspetti l'overloading è simile al polimorfismo.
L'overloading, che letteralmente significa sovraccaricamento, consiste nella possibilità di definire più versioni di uno stesso metodo all'interno della stessa classe, utilizzando lo stesso nome ma un numero e/o un tipo diverso di argomenti. In fase di esecuzione, sulla base degli argomenti effettivamente passati alla routine, verrà richiamata la procedura corretta; nella Guida in linea di Visual Basic sono specificati con precisione i passi che il compilatore esegue per risolvere l'overloading, cioè per determinare quale sia la routine corretta da richiamare. Nei prossimi articoli vedremo esempi dettagliati dell'uso del polimorfismo, ereditarietà ed overloading.

CREAZIONE DI UNA CLASSE

Applichiamo ora alcuni concetti della OOP appena descritti e vediamo come creare una classe in VB.NET. Per inserire una nuova classe in un progetto si deve:

- Selezionare dal menu *Progetto* la voce *Aggiungi classe*, verrà mostrata la finestra di dialogo *Aggiungi nuovo elemento*, con il modello *Classe* selezionato (nella parte destra della maschera).

Imitazione

L'evento *Inizializze*, generato quando si crea un'istanza della classe in VB6, non è più presente in VB .Net per contro in VB .Net si può definire un costruttore della classe, ossia un metodo che viene automaticamente richiamato quando si crea un'istanza della classe stessa. Contrariamente a quanto avveniva in VB6 mediante il costruttore è possibile specificare le proprietà iniziali della classe. Il costruttore è una particolare Sub il cui nome deve obbligatoriamente essere New.

- Inserire il nome della classe. Per il nostro esempio scrivere: *cliente*.

- Cliccare sul pulsante *Apri*. Si aprirà la finestra di progettazione del codice di VB.NET con il cursore all'interno della dichiarazione della classe:

```
Public Class Cliente
End Class
```

Nella finestra del codice si potranno scrivere le proprietà ed i metodi della classe appena creata.
Una tipica classe consiste di tre parti:

- dichiarazione delle variabili che utilizzerà soltanto la classe

- dichiarazioni delle proprietà

- implementazione dei metodi ossia le procedure o funzioni che gestiscono le variabili e le proprietà.

LE PROPRIETÀ

Per definire una proprietà in una classe il modo più semplice è quello di dichiarare una variabile pubblica nel modulo di classe. Questo metodo richiede una riga di codice per ogni proprietà.
Supponendo di voler implementare la classe cliente:

```
Public Nome As String
Public Cognome As String
Public SpesaMensile As Decimal
```

La parola chiave *Public* rende visibile la variabile da qualsiasi parte del codice che utilizza l'oggetto. Anche se questo è il metodo più semplice per definire le proprietà in una classe, non è il metodo raccomandato poiché verrebbe a cadere il requisito dell'incapsulamento che abbiamo già visto essere un requisito fondamentale della OOP.

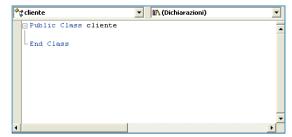



Fig. 1: Dichiarazione della classe Cliente.

Per evitare i rischi di variabili *Public*, le variabili che definiscono le proprietà della classe si dovranno dichiarare *Private*.

```
Private mvarNome As String
Private mvarCognome As String
Private mvarSpesaMensile As Decimal
```

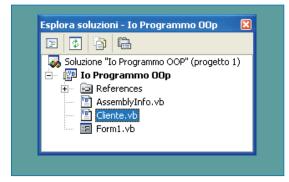



Fig. 2: In "Esplora soluzioni" è evidenziata la classe Cliente.VB.

Per dichiarare variabili visibili soltanto all'interno della classe è possibile utilizzare anche la classica istruzione *Dim*. Per convenzione e bene far precedere le variabili private che dovranno definire le proprietà della classe da un prefisso, nel nostro esempio il prefisso *mvar*, per distinguerle dalle proprietà che la classe renderà disponibile. Soltanto le procedure all'interno del modulo di classe potranno quindi modificare il valore di queste variabili.

LE PROCEDURE PROPERTY

A questo punto la domanda diventa: se le proprietà devono essere dichiarate *Private* e pertanto non possono essere modificate dall'esterno, a cosa servono? Per questo ci viene in aiuto il metodo *Property*. Per chi ha esperienza di VB6 il metodo *Property* ha subito notevoli modifiche, infatti al posto delle usuali *Property Set*, *Property Let*, *Property Get*, usate rispettivamente per assegnare e recuperare il valore di una proprietà, è stato tutto riunito in un unico metodo, che include sia il codice per recuperare sia quello per impostare il valore di una proprietà.
In pratica all'interno di un blocco *Property ... end Property* si inserisce:

- il costrutto *Get. ..End Get* che permette di ottenere il valore di una proprietà da altre parti del codice. Il codice all'interno del costrutto dovrà preoccuparsi di restituire il valore della proprietà richiesta. Si può includere al suo interno qualsiasi altro codice (ad es. calcoli oppure conversione di dati).

- il costrutto *Set...End Set* che permette di impostare il valore di una proprietà da altre parti del codice. Anche in questo costrutto si può scrivere qualsiasi altro codice, ad esempio per la convalida e conversione dei dati o per assegnare un valore di default alla proprietà.

Nel nostro esempio, per definire la proprietà *Nome*, dovremo scrivere:

```
Property Nome() As String
    Get
```

Glossario riassuntivo della OOP

Oggetto: un insieme correlato di dati e funzioni

Attributo o Proprietà: una singola informazione di dettaglio dell'oggetto

Metodo: un'azione che l'oggetto può compiere

Classe: un insieme di oggetti con gli stessi attributi e metodi

Incapsulamento: legame tra metodi e attributi di una classe, per la protezione da accessi esterni non controllati

Ereditarietà: derivazione delle proprietà di una classe da una classe di base predefinita

Polimorfismo: comportamento differenziato della stessa azione

Overloading: definizione di più versioni di uno stesso metodo all'interno di una classe

Client: classe che richiede un servizio

Server: classe che fornisce un servizio.

```
            Return mvarNome
        End Get

        Set(ByVal Valore As String)
            mvarNome = Valore
        End Set
    End Property
```

Il codice all'interno di *Set* verrà eseguito ogni volta che si assegna un valore alla proprietà *Nome*. Il valore della proprietà viene passato come parametro alla procedura per valorizzare la variabile interna *mvarNome*. Scrivendo:

```
ObjCliente.Nome = "Luigi"
```

viene valorizzata la proprietà *Nome* dell'oggetto *ObjCliente* pari a *Luigi*.
La sintassi del costrutto *Get* è simile a quella di una funzione che restituisce un valore dello stesso tipo della proprietà.
Scrivendo:

```
TextBox1.Text = ObjCliente.Nome
```

Viene chiamato il codice all'interno del costrutto *Get* che restituisce il valore della variabile privata *mvarNome*, per questo nel TextBox verrà visualizzata la stringa *Luigi*.
Analogamente per le altre proprietà dovremo scrivere:

```
    Property Cognome() As String
        Get
            Return mvarCognome
        End Get

        Set(ByVal Valore As String)
            mvarCognome = Valore
        End Set
    End Property

    Property SpesaMensile() As Decimal
        Get
            Return mvarSpesaMensile
        End Get

        Set(ByVal Valore As Decimal)
            'per evitare valori negativi
            If Valore < 0 Then
                Valore = 0
            End If
            mvarSpesaMensile = Valore
        End Set
    End Property
```

Nel costrutto *Set* di *SpesaMensile* è stato inserito un esempio di convalida dei dati con un controllo, tramite un'istruzione *If..Then*, che eviti la possibilità di assegnare un valore negativo alla proprietà *SpesaMensile*.

In relazione alla scrittura dei costrutti *Get. ..End Get* ed *Set...End Set*, le proprietà possono essere:

- Proprietà di sola lettura. Per creare una proprietà di sola lettura, è sufficiente omettere il costrutto *Set*.
- Proprietà di sola scrittura. Per creare una proprietà di sola lettura, è sufficiente omettere il costrutto *Get*.
- Proprietà accessibili in lettura e scrittura. Per creare una proprietà accessibile in lettura e scrittura, è necessario implementare i due costrutti *Get* e *Set*.

METODI

Dopo aver definito le proprietà di un oggetto si potranno scrivere le procedure o le funzioni che potranno svolgere operazioni con i dati memorizzati nelle proprietà, tali procedure o funzioni sono dette *metodi*. Per definire un metodo è sufficiente quindi scrivere una *Sub* oppure una *Function* all'interno di una classe. L'implementazione di un metodo sarà nascosta ad altre parti dell'applicazione. Nella classe *cliente* del nostro esempio si potrà scrivere il seguente metodo per il calcolo della spesa annua:

```
    Public Function CalcolaStipendio() As Decimal
        CalcolaStipendio = SpesaMensile * 12
    End Function
```

Per definizione, un metodo deve essere dichiarato *Public* perché sia visibile all'esterno della classe, per scrivere, invece, una routine che viene utilizzata soltanto all'interno della classe e che non è di alcun utilizzo all'esterno di essa, si dovrà dichiarare la routine come *Private*.
Nella definizione di una routine è possibile usare anche la nuova parola chiave *Protected*. Dichiarando una routine di una classe come *Protected*, essa non sarà accessibile alle istanze della classe ma sarà ereditabile da eventuali classi derivate. In pratica, *Protected* equivale a *Private*, con la differenza che il metodo è visibile anche alle classi che ereditano da quella principale.
Nella scrittura del metodo si usano i nomi delle proprietà al posto dei nomi interni delle variabili, che sono comunque sempre visibili all'interno della classe, questo comportamento è sempre auspicabile poiché in questo caso viene sempre eseguito il codice all'interno della *Property*.

CONCLUSIONI

Con la programmazione ad oggetti e con un minimo impegno da parte del programmatore è ormai abbastanza facile produrre un codice riusabile e facilmente mantenibile, per questo non perdetevi d'animo e continuate a seguirci nei meandri della OOP.

Ing. Luigi Buono

Note

È possibile raggruppare insieme più classi all'interno di un *Namespace*. Ad esempio:

```
Namespace Test
  Class Test1
    '...
  End Class
  Class Test2
    '...
  End Class
End Namespace
```

Con questo costrutto, per dichiarare un oggetto di tipo *Test1* è necessario scrivere:

```
Dim Prova1 As New
            Test.Test1()
```

Analogamente, per oggetti di tipo *Test2:*

```
Dim Prova1 As New
            Test.Test2()
```

Fatto questo, è possibile accedere ai metodi ed alle proprietà delle classi *Test1* e *Test2* nel modo consueto.
Un *namespace* è sempre *Public*, mentre i componenti all'interno possono essere *Public* o *Friend*. È inoltre possibile definire namespace nidificati.

Sovraccarico
DEGLI OPERATORI (PRIMA PARTE)

C#

Sovraccaricare gli operatori è una tecnica spesso utile, quando si elaborano nuovi tipi di dati. C#, in maniera semplice e flessibile, permette il sovraccarico degli operatori, analogamente a quanto avviene con C++. Questa lezione del corso introduce l'argomento e dimostra una prima serie di possibili operatori sovraccarichi.

Il sovraccarico degli operatori è una tecnica di programmazione non contemplata in Java, che C# riprende direttamente dall'esperienza di C++. Il CLR di .NET, dunque, permette l'adattamento dei più comuni tipi di operatori, in modo che il loro significato possa essere esteso ai tipi personalizzati, realizzati mediante le classi e le più comuni tecniche della programmazione orientata agli oggetti. In questa lezione discuteremo l'utilità e le tecniche di sovraccarico degli operatori, esaminando alcuni esempi dimostrativi.

DOVE, COME, QUANDO E PERCHÉ

Per avvicinare l'argomento della lezione, prendiamo in esame un esempio preliminare. Andiamo a considerare l'operatore di somma algebrica, ossia il classico +. Questo è tipicamente un operatore sovraccarico, che ricopre significati differenti secondo il contesto in cui viene applicato. Valutiamo il seguente codice:

```
int a = 5;
int b = 3;
int c = a + b;
```

In questo caso, l'operatore + esegue una somma algebrica tra due valori di tipo *int*. Diversamente avviene con altri tipi di dati:

```
double a = 4.12;
double b = 2.09;
double c = a + b;
```

Qui, la somma algebrica è svolta tra due valori double. Un caso ancora più particolare è il seguente:

```
string a = "ci";
string b = "ao";
string c = a + b;
```

Lavorando con le stringhe, l'operatore + esegue la concatenazione, dimenticando i precedenti significati di somma algebrica. In sostanza, l'operatore esaminato si comporta diversamente in base al contesto. Un unico operatore può svolgere più operazioni, in corrispondenza degli operandi coinvolti nell'espressione che lo comprende. Bene, C# permette di estendere ulteriormente i significati che il motore di esecuzione già attribuisce ai suoi operatori. Pertanto, è possibile creare un nuovo tipo di dati (tipicamente, una classe), che implementi in maniera differente le comuni operazioni ammesse dal linguaggio. Prendiamo in considerazione la classe *Vettore2D*:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) { i = ci; j = cj; } }
```

Vettore2D fornisce la rappresentazione di un comune vettore bidimensionale, costituito dall'unione delle due coordinate *i* e *j*. Chi ha studiato algebra sa che esiste l'operazione di somma vettoriale. C#, però, non lo sa. Provando il codice:

```
Vettore2D a = new Vettore2D(4, 3);
Vettore2D b = new Vettore2D(2, 1);
Vettore2D c = a + b;
```

si ottiene l'errore di compilazione riportato subito sotto:

```
error CS0019: Impossibile applicare l'operatore "+"
          a operandi di tipo "Vettore2D" e "Vettore2D".
```

File sul CD
\soft\codice\csharp12.zip

Sul Web
Tra le dispense Web del corso troverete appunti, aggiunte, approfondimenti, risposte a domande frequenti e altre appendici legate alla lezione odierna. L'indirizzo di riferimento è http://www.sauron-software.it/dispenseweb/csharp/

C#

In effetti, C# non ha la più pallida idea di come due valori di tipo *Vettore2D* possano essere sommati tra loro. Certo, nessuno gli ha detto come fare! Il sovraccarico degli operatori è una tecnica propria della programmazione orientata agli oggetti, che permette di informare il linguaggio su come interpretare gli operatori in presenza di nuovi tipi di dati. Sfruttando la tecnica, possiamo dotare C# della possibilità di sommare due *Vettori2D*. La somma di due vettori bidimensionali è così definita. Se $v1 = a1i + b1j$ e $v2 = a2i + b2j$ allora $v1 + v2 = (a1 + a2)i + (b1 + b2)j$. Preso atto di ciò, è facile dotare la classe *Vettore2D* di un nuovo metodo, capace di sommare due vettori bidimensionali, elaborando così un terzo valore:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) { i = ci; j = cj; }
  public static Vettore2D somma(Vettore2D a,
                                          Vettore2D b) {
    return new Vettore2D(a.i + b.i, a.j + b.j);} }
```

Secondo questa formulazione, la somma di due *Vettori2D* può essere svolta come nel seguente esempio:

```
Vettore2D a = new Vettore2D(4, 3);
Vettore2D b = new Vettore2D(2, 1);
Vettore2D c = Vettore2D.somma(a, b);
```

Questa tecnica è efficace, tuttavia non è un caso di sovraccarico di un operatore. Sarebbe molto più intuitivo e proficuo poter scrivere semplicemente:

```
Vettore2D c = a + b;
```

proprio come si era tentato inizialmente. Prendiamo in esame questa terza variante della classe *Vettore2D*:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) { i = ci; j = cj; }
  public static Vettore2D operator +(Vettore2D a,
                                          Vettore2D b) {
    return new Vettore2D(a.i + b.i, a.j + b.j); } }
```

La modifica effettuata è semplice: il metodo elaborato in precedenza è stato ribattezzato *operator +*. Tutto il resto è stato lasciato immutato. Ora è possibile scrivere ed eseguire quanto desiderato da principio, ossia:

```
Vettore2D a = new Vettore2D(4, 3);
Vettore2D b = new Vettore2D(2, 1);
Vettore2D c = a + b;
```

L'operatore +, in definitiva, è stato dotato di un nuovo significato, in corrispondenza della somma di due valori di tipo *Vettore2D*. Comodo ed utile, vero? Finora, per rendere semplice l'approccio all'argomento, siamo stati piuttosto pragmatici. Ora è il momento di esaminare più in teoria le norme di sovraccarico degli operatori. Per prima cosa, esistono operatori ed operatori. Nel corso delle lezioni precedenti, abbiamo fatto distinzione tra diverse categorie di operatori. Per avere una panoramica completa, è necessario rivisitare la classificazione attuata, esaminando il sovraccarico caso per caso.

SOVRACCARICO DEGLI OPERATORI ARITMETICI

Gli operatori aritmetici, sia binari sia unari, sono riportati di seguito:

```
+ - * / % ++ --
```

I primi cinque rispecchiano il modello già visto nel paragrafo precedente, che può essere così riassunto:

```
public static tipo-restituito operator +(tipo1 arg1, tipo2 arg2)
{ // ... }
public static tipo-restituito operator -(tipo1 arg1, tipo2 arg2)
{  // ... }
public static tipo-restituito operator *(tipo1 arg1, tipo2 arg2)
{  // ... }
public static tipo-restituito operator /(tipo1 arg1, tipo2 arg2)
{  // ... }
public static tipo-restituito operator %(tipo1 arg1, tipo2 arg2)
{  // ... }
```

Il valore restituito, di norma, è dello stesso tipo della classe che contiene il sovraccarico. Tuttavia, questo non è un obbligo. In casi specifici (ne vedremo uno a breve), si può agire diversamente. Per quanto riguarda gli argomenti in ingresso, è necessario che almeno uno dei due sia dello stesso tipo della classe che contiene il sovraccarico. L'altro, invece, non è vincolato. Quindi, è teoricamente possibile eseguire operazioni tra tipi differenti. Tornando al caso dei vettori bidimensionali, diviene possibile implementare il prodotto per uno scalare.
Se $v = ai + bj$ e n è un numerico qualsiasi, allora si definisce $v * n = (a * n)i + (b * n)j$.
Tradotto in C#:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) { i = ci; j = cj; }
  public static Vettore2D operator +(Vettore2D a,
                                          Vettore2D b) {
    return new Vettore2D(a.i + b.i, a.j + b.j); }
  public static Vettore2D operator *(Vettore2D v, int n) {
    return new Vettore2D(v.i * n, v.j * n); } }
```

Fatto ciò, è nelle nostre possibilità eseguire calcoli del tipo:

Conversione implicita

Le norme di conversione automatica esaminate nel corso delle lezioni precedenti possono essere applicate anche agli operatori sovraccaricati dal programmatore. Pertanto, una volta definito il prodotto tra un *Vettore2D* ed un *int*, diventa automaticamente possibile impiegare come secondo argomento qualsiasi dato implicitamente riconducibile ad un *int*. Ad esempio, un *byte* o uno *short*.

```
Vettore2D a = new Vettore2D(3, 5);
Vettore2D b = a * 5;
```

Se invece scriviamo

```
Vettore2D a = new Vettore2D(3, 5);
Vettore2D b = 5 * a;
```

otteniamo un errore di compilazione, perché l'operatore * non è automaticamente commutativo. Sta a noi decidere quando lo deve essere. Non c'è problema:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) { i = ci; j = cj; }
  public static Vettore2D operator +(Vettore2D a,
                                              Vettore2D b) {
    return new Vettore2D(a.i + b.i, a.j + b.j);
  }
  public static Vettore2D operator *(Vettore2D v, int n) {
    return new Vettore2D(v.i * n, v.j * n);
}
  public static Vettore2D operator *(int n, Vettore2D v) {
    return v * n;
  }
}
```

Poiché, nel nostro specifico caso, il prodotto tra un vettore ed un intero è commutabile, abbiamo aggiunto un nuovo sovraccarico che, per semplicità, si riconduce al caso precedente. Bel lavoro, Tex! Passiamo ora in rassegna un ulteriore esempio, in cui il tipo restituito non appartiene alla classe che stabilisce il sovraccarico. I vettori sono una cava di esempi utili. Prendiamo in considerazione il prodotto scalare, che da due vettori desume un valore numerico. Se $v1 = a1i + b1j$ e $v2 = a2i + b2j$ allora $v1 * v2 = (a1 * a2) + (b1 * b2)$.
In codice:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) { i = ci; j = cj; }
  public static Vettore2D operator +(Vettore2D a,
                                              Vettore2D b) {
    return new Vettore2D(a.i + b.i, a.j + b.j); }
  public static Vettore2D operator *(Vettore2D v, int n) {
    return new Vettore2D(v.i * n, v.j * n); }
  public static Vettore2D operator *(int n, Vettore2D v) {
    return v * n; }
  public static int operator *(Vettore2D a, Vettore2D b) {
    return (a.i * b.i) + (a.j * b.j); }
}
```

La classe è stata ora dotata di una terza forma dell'operatore *, che accetta due *Vettori2D* come operandi, restituendo un *int*. Tutto funziona correttamente, basta provare:

```
Vettore2D a = new Vettore2D(4, 3);
Vettore2D b = new Vettore2D(2, 1);
int c = a * b;
```

Riassumendo, è possibile assegnare diversi significati, nell'ambito di una sola classe, ad uno stesso operatore. Ovviamente, non devono esistere contrasti o contraddizioni tra una definizione e l'altra: ogni sovraccarico deve riguardare un caso distinto. Occhio, dunque, a non cadere in un inganno molto comune. Supponiamo di voler definire, ora, anche il prodotto vettoriale, che da due vettori ne desume un terzo. Non sarebbe possibile utilizzare di nuovo il simbolo *, poiché è già stato impiegato per il prodotto scalare, in corrispondenza di due argomenti di tipo *Vettore2D*. Il tipo restituito non rende differenti due metodi con lo stesso nome e con un'analoga lista di argomenti. Dunque, per il prodotto vettoriale bisognerebbe adottare un altro simbolo. Andiamo agli operatori unari. Consideriamo la versione unaria degli operatori + e -, che seguono i modelli:

```
public static tipo-restituito operator +(tipo arg) {  // ...}
public static tipo-restituito operator -(tipo arg) {  // ... }
```

L'unico argomento accettato deve essere un valore della classe che sovraccarica l'operatore. Riprendiamo *Vettore2D* e diamogli la possibilità di invertire uno dei suoi valori:

```
class Vettore2D {
  public int i;
  public int j;
  public Vettore2D(int ci, int cj) { i = ci; j = cj; }
  public static Vettore2D operator +(Vettore2D a,
                                              Vettore2D b) {
    return new Vettore2D(a.i + b.i, a.j + b.j); }
  public static Vettore2D operator *(Vettore2D v, int n) {
    return new Vettore2D(v.i * n, v.j * n); }
  public static Vettore2D operator *(int n, Vettore2D v) {
    return v * n; }
  public static int operator *(Vettore2D a, Vettore2D b) {
    return (a.i * b.i) + (a.j * b.j); }
  public static Vettore2D operator -(Vettore2D v) {
    return new Vettore2D(-v.i, -v.j); } }
```

Adesso possiamo calcolare qualcosa come:

```
Vettore2D a = new Vettore2D(4, 3);
Vettore2D b = -a;
```

Dato che siamo diventati bravi con gli operatori, definiamo ora anche la forma binaria dell'operatore -, che può essere ricondotta facilmente ad altri casi precedenti. Infatti: $v1 - v2 = v1 + (-v2)$. Quindi:

```
class Vettore2D {
  public int i;
  public int j;
```

C#

Bibliografia

• **GUIDA A C#**
Herbert Schildt
(McGraw-Hill)
ISBN 88-386-4264-8
2002

• **INTRODUZIONE A C#**
Eric Gunnerson
(Mondadori Informatica)
ISBN 88-8331-185-X
2001

• **C# GUIDA PER LO SVILUPPATORE**
***Simon Robinson* e altri**
(Hoepli)
ISBN 88-203-2962-X
2001

C#

Bibliografia

• **USER INTERFACES IN C#: WINDOWS FORMS AND CUSTOM CONTROLS**
Matthew MacDonald
(APress)
Pagine: 586 - $ 49,95
ISBN: 1-59059-045-7
www.apress.com

Questo volume combina, in forma unica, lo studio delle Windows Forms di .NET ed i principali criteri per un buon design delle interfacce utente. Piuttosto approfondito e ben fatto, il volume presenta esperienze pratiche di ottimo impatto, dimostrando ogni concetto preso in considerazione nel corso dello studio. I concetti di usabilità sono immediatamente presi in considerazione, sin dal primo capitolo. Un'interessante digressione è dedicata all'eterna guerra tra creatività ed osservanza delle convenzioni. Quindi, si passa alla pratica. Dopo una rapida carrellata sui pregi della programmazione orientata agli oggetti e alla sua implementazione con C#, è considerata la libreria Windows Forms, il fulcro per lo sviluppo di interfacce utente con .NET. Tutti gli elementi tipici delle interfacce utente vengono passati in rassegna. In un colpo solo, è dimostrato il loro uso con C# e sono dibattute le loro funzionalità in un contesto volto all'ottenimento di interfacce semplici, funzionali ed usabili. Un ottimo volume per chi intende scegliere C# come strumento per lo sviluppo di applicazioni Windows destinate all'utente finale.

```
 public Vettore2D(int ci, int cj) { i = ci; j = cj; }
 public static Vettore2D operator +(Vettore2D a,
                                        Vettore2D b) {
   return new Vettore2D(a.i + b.i, a.j + b.j); }
 public static Vettore2D operator *(Vettore2D v, int n) {
   return new Vettore2D(v.i * n, v.j * n); }
 public static Vettore2D operator *(int n, Vettore2D v) {
   return v * n;}
 public static int operator *(Vettore2D a, Vettore2D b) {
   return (a.i * b.i) + (a.j * b.j); }
 public static Vettore2D operator -(Vettore2D v)
 {  return new Vettore2D(-v.i, -v.j); }
 public static Vettore2D operator -(Vettore2D a,
                         Vettore2D b) { return a + (-b); }}
```

Concludiamo con gli operatori unari ++ e --, che seguono il modello:

```
public static tipo-restituito operator ++(tipo arg) {// ...}
public static tipo-restituito operator --(tipo arg) {// ...}
```

Come nel caso precedente, l'unico argomento accettato deve essere un valore della classe che sovraccarica l'operatore. Produciamo l'ennesima nuova versione di Vettore2D:

```
class Vettore2D {
 public int i;
 public int j;
 public Vettore2D(int ci, int cj) { i = ci; j = cj; }
 public static Vettore2D operator +(Vettore2D a,
                                        Vettore2D b) {
   return new Vettore2D(a.i + b.i, a.j + b.j);
 }
 public static Vettore2D operator *(Vettore2D v, int n) {
   return new Vettore2D(v.i * n, v.j * n);
 }
 public static Vettore2D operator *(int n, Vettore2D v) {
   return v * n;
 }
 public static int operator *(Vettore2D a, Vettore2D b) {
   return (a.i * b.i) + (a.j * b.j);
 }
 public static Vettore2D operator -(Vettore2D v) {
   return new Vettore2D(-v.i, -v.j);
 }
 public static Vettore2D operator -(Vettore2D a,
                                        Vettore2D b) {
   return a + (-b); }
 public static Vettore2D operator ++(Vettore2D v) {
   return new Vettore2D(v.i++, v.j++);
 }
 public static Vettore2D operator --(Vettore2D v) {
   return new Vettore2D(v.i--, v.j--);
  }
}
```

Questi due nuovi sovraccarichi sembrano funzionare, se si scrive qualcosa come:

```
Vettore2D a = new Vettore2D(4, 3);
Vettore2D b = a++;
System.Console.WriteLine("b = " + b.i + "i + " + b.j + "j");
```

Tuttavia, gli operatori ++ e --, secondo la loro definizione canonica, dovrebbero influire anche sull'oggetto sul quale vengono richiamati, alterandone i contenuti. In questo caso, invece, il contenuto di *a* non è stato variato. Lo si può verificare:

```
Vettore2D a = new Vettore2D(4, 3);
Vettore2D b = a++;
System.Console.WriteLine("a = " + a.i + "i + " + a.j
                                              + "j");
System.Console.WriteLine("b = " + b.i + "i + " + b.j + "j");
```

Insomma, *b* va bene ma *a* no. Sintatticamente, quello che abbiamo fatto è corretto, ma in realtà non lo è logicamente. Se vogliamo rientrare nella prassi, dobbiamo formulare diversamente il corpo dei due sovraccarichi:

```
class Vettore2D {
 public int i;
 public int j;
 public Vettore2D(int ci, int cj) { i = ci; j = cj;}
 public static Vettore2D operator +(Vettore2D a,
                                        Vettore2D b) {
   return new Vettore2D(a.i + b.i, a.j + b.j);}
 public static Vettore2D operator *(Vettore2D v, int n) {
   return new Vettore2D(v.i * n, v.j * n);}
 public static Vettore2D operator *(int n, Vettore2D v) {
   return v * n; }
 public static int operator *(Vettore2D a, Vettore2D b) {
   return (a.i * b.i) + (a.j * b.j);}
 public static Vettore2D operator -(Vettore2D v) {
   return new Vettore2D(-v.i, -v.j); }
 public static Vettore2D operator -(Vettore2D a,
                         Vettore2D b) {return a + (-b);
 }
 public static Vettore2D operator ++(Vettore2D v)
 { v.i++; v.j++; return v; }
 public static Vettore2D operator --(Vettore2D v)
 {
    v.i--; v.j--; return v;
 }
}
```

Ecco, ora ci siamo. Ricordatevi sempre di questo esempio quando andrete a sovraccaricare gli operatori ++ e --.

CONCLUSIONI

Per oggi, ci fermiamo qui. Nel corso delle prossime lezioni passeremo in rassegna gli altri operatori di C#, discutendo nel dettaglio le opportune tecniche di sovraccarico da intraprendere.

Carlo Pelliccia

Classi base ASTRATTE

Come promesso, entriamo in questa lezione nell'ambito di quelle funzionalità che fanno del C++ un linguaggio eccellente per la programmazione a oggetti. Vedremo in che modo paroloni come "funzione virtuale pura" e "classe base astratta" siano in realtà concetti molto utili nello sviluppo di applicazioni che fanno uso del meccanismo dell'ereditarietà.

Nella precedente lezione, abbiamo mostrato come operano alcuni dei meccanismi più interessanti e utili che il C++ mette a disposizione nel campo dell'ereditarietà. Abbiamo in particolar modo visto l'utilizzo delle funzioni virtuali, che sono le funzioni che consentono di realizzare il cosiddetto late binding, ovvero la decisione a tempo di esecuzione di quale deve essere la funzione da utilizzare per una determinata chiamata. In particolare esse permettono di decidere se deve essere invocata la funzione dell'oggetto base o la funzione dell'oggetto derivato, ridefinita tramite overload. Come abbiamo visto, il late binding ha un aspetto positivo, che è quello di potere utilizzare sempre la funzione giusta, e purtroppo anche un aspetto negativo, che è quello dell'inefficienza di demandare a tempo di esecuzione la decisione di quale funzione invocare. L'aspetto negativo purtroppo non può essere eliminato (in quanto è l'essenza stessa del late binding) ma si può tentare di ottimizzare almeno la generazione del codice, evitando di compilare quelle funzioni della classe base che già si sa che verranno usate solo come base per un successivo overload. Per fare questo bisogna usare il meccanismo delle funzioni virtuali pure. Le funzioni virtuali pure (FVP) sono delle particolari funzioni virtuali che hanno la caratteristica di non dovere essere definite, cioè non bisogna scrivere il codice che le implementa, in quanto esse sono soltanto una specie di "segnalibro" per i futuri overload. Una FVP si dichiara nel seguente modo:

```
virtual <tipo-restituito> <nome-funzione>
                (<lista-parametri>) = 0;
```

cioè esattamente allo stesso modo di una funzione virtuale ordinaria (si usa la parola chiave "virtual") ma con un "= 0" postfisso. L'utilizzo tipico che si fa di una FVP può essere esemplificato nel seguente codice:

```
class Base {
public:
        virtual void stampa() = 0;
//... altre definizioni qui... };
class Derivata : public Base {
public:
   void stampa() { cout << "ciao!";}
   //... altre definizioni qui... };
//... all'interno del codice...
Derivata d;
d.stampa();
```

La FVP stampa() di Base non è implementata da nessuna parte, mentre è implementata quella della sua classe derivata Derivata, che è effettivamente quella utilizzata nelle chiamate. Una cosa molto importante da notare è che il seguente codice:

```
Base b;
b.stampa();
```

genera un errore a tempo di compilazione. Il lettore più attento magari lo avrà intuito, tuttavia il motivo di questo errore non è quello che ci si potrebbe aspettare, ma una cosa più sottile. L'errore si presenta non perché si sta cercando di richiamare una funzione il cui codice non è stato implementato (seconda istruzione), quanto piuttosto perché si sta cercando di istanziare un oggetto di una classe base astratta (prima istruzione). In C++ è detta classe base astratta (CBA) qualsiasi classe che contenga almeno una FVP. L'errore nel codice precedente sta quindi nell'avere scritto:

File sul CD
\\soft\codice\CodiceC.zip

FVP e funzioni vuote

È essenziale non confondere una FVP con una funzione vuota (che cioè non contiene istruzioni): per la funzione vuota viene generato un codice che non fa nulla, mentre per la FVP il codice non viene proprio generato; e infatti non si può invocare direttamente una FVP mentre si può tranquillamente chiamare una funzione vuota, nonostante questa non abbia alcun effetto sull'esecuzione del programma.

C++

```
Base b;
```

Il compilatore C++ vieta di istanziare un oggetto di una CBA, proprio perché questa contiene delle funzioni (le FVP) che sono state inserite unicamente al fine di derivare la classe in questione, ed evitare così che essa sia utilizzata "da sola". Questo comportamento porta all'apparente paradosso che anche una funzione della classe base, che non sia una FVP, non possa essere comunque utilizzata. In altre parole se Base fosse definita in questo modo:

```
class Base {
public:
  virtual void stampa() = 0;
  void stampa2() {cout << "Non sono una FVP!";}
//... altre definizioni qui... };
```

il seguente codice:

```
Base b2;
b2.stampa2();
```

che all'apparenza ha tutti gli elementi per funzionare, in realtà genererebbe il medesimo errore precedente. Se invece avessimo le istruzioni:

```
Derivata d2;
d2.stampa2();
```

esse produrrebbero effettivamente la stampa a schermo desiderata, nonostante la definizione di Derivata non aggiunga nulla alla funzione stampa2(), che è l'unica usata. Precludere anche l'utilizzo di funzioni definite in una CBA può sembrare un tantino drastico, ma in realtà la situazione è meno problematica di quanto sembri, principalmente per due motivi:

1. Nessuno vi obbliga a utilizzare CBA e FVP se non lo volete...
2. Qualora davvero decidete di utilizzare CBA e FVP, è proprio perché volete ottenere questo tipo di comportamento, quindi si tratta di una scelta consapevole.

Come già accennato, l'utilità delle FVP è legata alla struttura che si intende dare al codice. Utilizzando le CBA ci possiamo garantire che tutte le classi derivate, sicuramente dovranno implementare le FVP che dichiariamo, per essere utilizzate. E vedremo tra poco come questa sia una bella certezza.

UN PO' DI TEORIA

A ben vedere, potrebbe apparire oscuro il motivo per cui si dovrebbe obbligare un utente di una nostra classe a ridefinire funzioni, secondo quelli che sono i nostri desideri: potrebbe venirci più spontaneo pensare di fornire una classe, lasciando agli altri la massima libertà. In realtà, il motivo di una scelta siffatta sarà presto evidente e giustificato. Parlammo già alcune puntate fa delle possibili relazioni tra oggetti (rif. "ioProgrammo" n.65, box pg.78), dicendo che esse possono essere, essenzialmente, di tre tipi: utilizzo (usa), contenimento (has-a) ed ereditarietà (is-a). E' proprio riguardo l'ereditarietà che si è detto che una relazione del tipo "A is-a B" significa che l'oggetto di tipo A è anche (is-a) un oggetto di tipo B, e in questo senso è possibile usare (dietro l'ipotesi di derivazione public) un oggetto di tipo A al posto di un oggetto di tipo B (ma non viceversa). È proprio pensando ai possibili usi dell'ereditarietà che si parla delle CBA. Si pensi ad una CBA in cui tutti i metodi sono virtuali puri: in questa condizione, una classe da essa derivata deve necessariamente ridefinirli tutti, per potere essere istanziata. In realtà in una situazione di questo tipo, per la CBA sarebbe più consono il nome di interfaccia. Infatti, una interfaccia è un insieme di strumenti che permettono la comunicazione tra due "strati"; ad esempio, l'interfaccia utente di un programma permette, attraverso gli elementi che la compongono, la comunicazione tra l'utente ed il programma. Vediamo in che modo una CBA si può considerare una entità "di comunicazione". Abbiamo detto che non si possono creare istanze di oggetti appartenenti ad una CBA; tuttavia, una CBA può essere usata senza alcun problema per implementare altre classi o funzioni. Ad esempio, il seguente codice è corretto:

```
class Virtuale {
public:
  virtual void Saluta() = 0;
  virtual void IniziaDa(int) = 0;
  virtual int QuanteVolte() = 0;
  virtual void Successivo() = 0;};
void CicloFor(Virtuale& v)
{
  for(v.IniziaDa(10);
      v.QuanteVolte()>0;
      v.Successivo())
    v.Saluta();}
```

Possiamo quindi passare, come parametro di una funzione, un oggetto del tipo di una CBA (per riferimento: se passassimo l'oggetto per valore, si avrebbe una chiamata al costruttore di copia, che quindi andrebbe dichiarato come virtuale nella classe base). In questo modo, derivando una classe da una CBA, potremmo tranquillamente usare la funzione in questione (ad es. CicloFor()) sul nuovo oggetto e questa funzione risulterebbe scritta una volta per tutte, rendendo così il codice altamente riutilizzabile. Sfruttando questo meccanismo, la CBA fa da interfaccia tra le sue classi derivate e le funzioni in cui essa viene utilizzata, permettendo la

Liste collegate

Una lista collegata è una classica struttura dati, spesso usata in informatica, che viene implementata tramite l'utilizzo di puntatori. Essa rappresenta un insieme di elementi ordinati dello stesso tipo, sulla quale sono definite particolari funzioni caratteristiche (ad es. Inserisci() o Rimuovi()). Definendo funzioni che effettuano in maniera opportuna inserimenti e rimozioni, si può usare una lista collegata per implementare altre strutture, come pile (in cui viene inserito e rimosso sempre il primo elemento) o code (in cui viene inserito un elemento sempre all'ultimo posto e rimosso sempre quello al primo posto).

comunicazione tra loro.

UN ESEMPIO (CLASSICO)

Immaginiamo di voler costruire una funzione che si occupi di calcolare il perimetro di un poligono. A ben vedere, sembra un problema banale: basta passare come parametri i valori delle lunghezze dei lati, e il gioco è fatto. In effetti si potrebbe implementare una funzione di questo tipo:

```
//struttura di appoggio
struct elem
{ int lato;
  elem* next; };
int CalcolaPerimetro(elem* primo_lato)
{ //primo_lato rappresenta un puntatore
  //all'inizio di una catena di elementi
  int perimetro = 0;
  elem* p = primo_lato;
  while(p!=NULL) {
    //finche' c'e' ancora un lato da sommare...
    perimetro += p->lato;
    p = p->next; }
  return perimetro; }
```

Questa funzione prende una catena di elementi (detta lista collegata) di tipo elem, e la scorre sommando tutti i valori del campo lato, ottenendo in questo modo il perimetro cercato. Il vantaggio di passare una lista collegata risiede nel fatto che non bisogna specificare il numero di lati del poligono. Questa funzione svolge il suo compito egregiamente, tuttavia ha un difetto: il suo funzionamento si basa su una particolare struttura dati, che il programmatore è tenuto conoscere e utilizzare in maniera alquanto rigida. Proviamo a ragionare in modo diverso. Iniziamo col pensare ad una classe Poligono, la quale rappresenta tutti i possibili poligoni di N lati (con N variabile), e che potrebbe essere così definita:

```
class Poligono
{
public:
virtual int QuantiLati() = 0;
virtual int LunghezzaLato(int) = 0; };
```

Se la definiamo come CBA, non possiamo usarla direttamente. Per poterla usare, dovremo definire delle classi da essa derivate, ad esempio:

```
class Quadrato: public Poligono
{ //codice della classe };
class Rettangolo: public Poligono
{  //codice della classe };
```

e così via. In questo modo, come conseguenza dell'ereditarietà, un oggetto della classe derivata è anche un oggetto della classe base (e quindi, ad esempio un Rettangolo è anche un Poligono). Proviamo a pensare ad una funzione come la seguente:

```
int CalcolaPerimetro(Poligono& p)
{ //codice... ??? }
```

Noi non possiamo dichiarare oggetti di tipo Poligono, ma possiamo dichiarare oggetti di altri tipi compatibili col tipo Poligono. Ad esempio sarebbe corretto scrivere:

```
Quadrato quad(7);
//quadrato di lato 7
Rettangolo rett(5,10);
//due lati lunghi 5 e due lunghi 10
cout << CalcolaPerimetro(quad) << "\n";
cout << CalcolaPerimetro(rett) << "\n";
```

A questo punto manca il codice della funzione CalcolaPerimetro() che dovrà implementare l'algoritmo di calcolo del perimetro, dato un oggetto derivato da Poligono; la funzione potrebbe essere così strutturata:

```
int CalcolaPerimetro(Poligono& p)
{
  int perimetro = 0;
  for(int i=1;i<=p.QuantiLati();i++)
    perimetro += p.LunghezzaLato(i);
  return perimetro;
}
```

Come facilmente si può osservare, la funzione appena scritta non usa alcuna struttura dati particolare: per essere usata correttamente, basta passarle un oggetto derivato da Poligono, che implementi le funzioni QuantiLati() e LunghezzaLato() col giusto significato. A questo punto il cerchio si chiude (anche se parliamo di poligoni): chiunque voglia usare la nostra funzione per il calcolo dei perimetri, deve per forza derivare la classe Poligono, e il fatto che questa sia una CBA ci dà la certezza che tutte le sue FVP vengano implementate negli oggetti derivati.
Ovviamente la correttezza del comportamento delle FVP è demandata al programmatore.

CONCLUSIONI

Gli argomenti del C++ che stiamo trattando in queste puntate sono tra i più utili, poiché di carattere generale. La loro conoscenza è preziosa e può essere tranquillamente "riciclata" nell'apprendimento di altri linguaggi di tipo OO. Nella prossima puntata vedremo come aumentare ulteriormente la quantità di codice riutilizzabile in nostro possesso: non mancate!

Alfredo Marroccelli e Marco Del Gobbo

Contatta gli autori!

Se hai suggerimenti, critiche, dubbi o perplessità sugli argomenti trattati e vuoi proporli agli autori puoi scrivere agli indirizzi:

alfredo.marroccelli@libero.it **(Alfredo)**
e
marcodelgobbo@libero.it **(Marco).**

Questo contribuirà sicuramente a migliorare il lavoro di stesura delle prossime puntate.

MATLAB®

IL LINGUAGGIO DEL CALCOLO NUMERICO PER LE APPLICAZIONI TECNOLOGICHE

MATLAB

Siamo al primo appuntamento con un ambiente che é divenuto il linguaggio standard nel calcolo numerico. È uno strumento affidabile e adatto ad applicazioni nell'ambito della tecnologia e della ricerca scientifica.

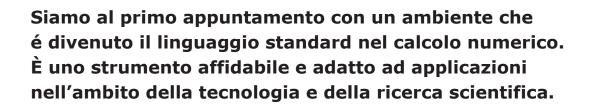
File sul CD
\soft\codice\matlab.zip

MATLAB® ha come suo scopo principale quello di offrire capacità di calcolo e simulazione in un ambiente semplice da usare e completo. Il linguaggio possiede funzionalità che spaziano da quelle algoritmiche, a quelle per la manipolazione di grandi data set, a quelle di visualizzazione di alta qualità, a quelle di creazione di interfacce utente. L'integrazione di tutte queste componenti è di estrema utilità poiché ci consente di fare evolvere le nostre idee in un ambiente omogeneo. Siamo così in grado di trarre tutti i possibili benefici dall'uso di una grande varietà di algoritmi frutto del lavoro di ricerca in un vasto numero di discipline scientifiche differenti. Nell'industria e nella ricerca moderna vi è necessità di produttività e di performance al fine di rilasciare soluzioni innovative in tempi compatibili con la complessità e la velocità con cui le nuove tecnologie appaiono sul mercato. Ci si trova di fronte con sempre maggiore frequenza ad applicazioni che hanno una intrinseca complessità che va oltre il livello che si è in grado di gestire in maniera produttiva con strumenti tradizionali. Per ovviare a questo problema è necessario possedere uno strumento che consenta di eseguire analisi, simulazioni e test in tempi brevi. Analizzare dati significa, in ultima analisi, poter trattare grandi massi di dati e applicare loro algoritmi che siano in grado di manipolare le informazioni ad alta velocità. Mentre, la simulazione ci consente di costruire un modello virtuale del fenomeno, di esplorare soluzioni alternative, studiare comportamenti, stabilire strategie di analisi e di test; in altri termini, possiamo trattare sistemi più complessi. Il test nelle applicazioni della moderna tecnologia ha assunto un'importanza rilevante. La possibilità di fare molte prove in un ambiente virtuale ci consente di individuare la migliore soluzione. Sovente si opta e ci si accontenta della soluzione che dà più sicurezza. Non si ha il tempo ed il modo di ricercare la soluzione migliore poiché non si ha la possibilità di investire ulteriore tempo e risorse. MATLAB è specificatamente progettato per questi scopi. E' un linguaggio basato sull'algebra matriciale adatto sia all'uso

Fig. 1: **L'ambiente MATLAB durante l'utilizzo interattivo.**

MATLAB

interattivo sia a quello procedurale. Proprio l'uso interattivo ci viene in soccorso quando abbiamo la necessità di esplorare un concetto ma non vogliamo impegnarci nello sforzo di sviluppo di un programma. Sarà proprio questo il modo che adotteremo in questo primo tentativo di entrare in contatto con MATLAB. I concetti matematici che incontreremo verranno spiegati in maniera piana e divulgativa, usando quel minimo di rigore che il problema richiede rimandando ogni volta a testi specifici per chi desideri approfondire e apprendere le sottigliezze e le implicazioni. Quando lanciamo MATLAB ci troviamo di fronte ad una finestra simile a quella riportata in Fig.1. La finestra che ha come titolo *"Command Window"* è quella su cui andremo a digitare i nostri comandi. Cominciamo con il definire alcuni concetti fondamentali che sono indispensabili per districarsi tra i problemi matematici e comprendere il codice che scriveremo.

SCALARI, VETTORI E MATRICI

Chiamiamo *scalare* quella minuscola matrice, la più semplice, formata da una sola riga ed una sola colonna. Definirlo in MATLAB è un'operazione che ricorda quella usata in molti altri linguaggi:

```
>> a = 5
a =
    5
```

Notiamo che non abbiamo avuto la necessità di definire in anticipo quale dimensione debba avere *"a"*; MATLAB è in grado di definirla correttamente in maniera autonoma ed è in grado di espanderla automaticamente e di ampliarla a piacere. I vettori, invece, non sono altro che matrici con una sola delle dimensioni pari ad 1.

[1]

```
>> x = [1 2 3 4 a]
x =
    1    2    3    4    5
```

Abbiamo creato un vettore riga di cinque elementi usando come quinto elemento *"a"*.
Qualora desiderassimo un vettore colonna che contenesse gli stessi elementi di *"x"*, potremmo usare un operatore matematico che traspone (scambia le righe con le colonne, la prima riga diviene la prima colonna e così via); questo operatore è un semplice apice da far seguire al nome della variabile:

[2]

```
>> x'
ans =
    1
    2
    3
    4
    5
```

Possiamo creare la nostra variabile in una maniera ancora più veloce e concisa:

```
>> x = [1:5]
x =
    1    2    3    4    5
```

L'operatore ":" ci consente di generare automaticamente una sequenza di numeri. La sua forma più generale è:

```
>> x = [3:5:33]
x =
    3    8    13    18    23    28    33
```

Vengono generati numeri a partire da 3 che, con passo 5, raggiungono il valore 33. Dobbiamo ricordare che in MATLAB questo tipo di generazione viene utilizzato tutte le volte che ve ne sia la necessità; lo incontreremo tra breve nella generazione di indici per l'estrazione di sottomatrici così come nella determinazione degli indici dei cicli. Sovente abbiamo la necessità di eseguire operazioni su vettori. Se volessimo sommarli potremmo operare così:

```
>> x + x
ans =
    2    4    6    8    10
```

Per vettori di dimensioni compatibili (identiche), sommiamo gli elementi che si trovano nella stessa posizione. È di estrema importanza notare che ciò che abbiamo appena eseguito è un'operazione aritmetica che è del tutto assimilabile al calcolo di un'espressione matematica tra scalari. In MATLAB possiamo calcolare espressioni complesse che coinvolgono vettori e matrici di grandi dimensioni, utilizzando direttamente l'espressione matematica senza fare ricorso a cicli di calcolo. Il codice così prodotto è immediatamente comprensibile poiché rispetta pienamente la notazione matematica standard e si mantiene snello non dovendo ricorrere ad uno stile di scrittura pesante e soggetto ad errori. Se volessimo effettuare una moltiplicazione le cose si complicano: è possibile eseguire la moltiplicazione in due modi differenti. Il primo modo è quello che viene chiamato prodotto scalare:

```
>> x * x'
ans =
   55
```

Moltiplichiamo elemento per elemento e poi sommiamo tutti i numeri così ottenuti. Notiamo che per ottenere una perfetta congruenza delle dimen-

[1] Simboli

Le parentesi quadre delimitano la definizione di vettori e matrici. Spazi o virgole separano le colonne.

[2] Ans

La variabile *ans* è riservata ed il suo valore viene assegnato tutte le volte in cui non vi sia un'assegnazione esplicita del risultato ad una nuova variabile.

sioni abbiamo dovuto trasporre il secondo vettore. In effetti il prodotto eseguito così non è che un caso particolare di prodotto matriciale che vedremo tra breve. Il secondo modo viene definito come la moltiplicazione di ogni elemento del primo vettore per quello nella posizione corrispondente del secondo. Dobbiamo utilizzare un operatore specifico che ci consente di distinguere il tipo di moltiplicazione:

[3]

```
>> x .* x
ans =
   1    4    9    16    25
```

PRODOTTO MATRICIALE

Le matrici sono entità matematiche che possono essere rappresentate come tabelle rettangolari di numeri (i vettori e gli scalari non sono altro che matrici particolari). Tra loro interagiscono e possono essere messe in relazione per mezzo di vari operatori. Per esempio, per sommare e sottrarre due matrici tra loro è necessario che le loro dimensioni siano identiche:

[4]

```
>> A = [1 2 3; 4 5 6; 7 8 9]
A =
1          2          3
4          5          6
7          8          9
>> B = [10 11 12; 13 14 15; 16 17 18]
B =
10         11         12
13         14         15
16         17         18
>> A + B
ans =
11         13         15
17         19         21
23         25         27
```

Più complicata è l'operazione di moltiplicazione. Qui viene definita come prodotto righe per colonne: cioè ogni riga della prima matrice viene moltiplicata secondo un prodotto scalare per ogni colonna della seconda.

```
>> A * B
ans =
84         90         96
201        216        231
318        342        366
```

Se per esempio prendiamo la prima riga di *A [1 2 3]* e la prima colonna di *B [10 13 16]* e applichiamo un prodotto scalare tra questi due vettori eseguiremo l'operazione:

```
>> 1*10 + 2*13 + 3*16
ans =
   84
```

Essendo il frutto del prodotto scalare della riga numero 1 di *A* con la colonna numero 1 di *B* esso occuperà la posizione 1,1 nella matrice risultato. Se continuiamo di questo passo otteniamo la matrice che appare sopra. Qui vale la pena di fermarci un attimo per riflettere su un problema: non è possibile moltiplicare qualunque matrice per qualunque altra matrice. Infatti se seguiamo il procedimento analitico illustrato sopra, ci accorgiamo rapidamente che se il numero di colonne di *A* fosse differente dal numero di righe di *B* ci troveremmo di fronte ad una moltiplicazione di due vettori di lunghezze differenti e questo non è possibile. Dunque, il numero di colonne della prima matrice deve necessariamente essere identico al numero di righe della seconda matrice e, se questo è verificato, la matrice risultante avrà tante righe quante sono le righe della prima matrice e tante colonne quante sono le colonne della seconda. Da questo deriva il fatto che (a meno di avere a che fare con matrici quadrate) se è possibile eseguire *A*B* non è possibile eseguire il calcolo *B*A*:

```
>> A = [1 2 3; 4 5 6]
A =
1          2          3
4          5          6
>> A * B
ans =
84         90         96
201        216        231
>> B * A
??? Error using ==> *
Inner matrix dimensions must agree.
```

VISUALIZZAZIONE

Con questo fondamentale bagaglio possiamo ora affrontare problemi più interessanti e complessi. Ipotizziamo di possedere i dati di andamento dei prezzi alla produzione dei prodotti industriali per settore di attività economica e di volerli visualizzare per mezzo di un tipo di grafico (a coni) adottato in alcune applicazioni di carattere economico e finanziario. Iniziamo ad utilizzare MATLAB in maniera interattiva poiché abbiamo bisogno di verificare immediatamente l'andamento dei nostri ragionamenti e vogliamo avere l'opportunità di cambiare il corso dei pensieri mano a mano che i dati si trasformano davanti ai nostri occhi. Definiamo quindi i dati oggetto della nostra analisi.

[5]

```
>> dati = [1996   100.7   101.8   102.6   102.2   102.7
                          101.2   97.2   105.3   98.1;...
          1997   106.0   102.6   102.9   103.1   104.4
                         101.3   96.9   106.5   100.1;...
```

MATLAB

[3] .*

Il comando ".*" indica che si vuole effettuare un'operazione elemento per elemento.

[4] Punti e virgola ";"

I punti e virgola separano le righe.

[5] ...

Il punto e virgola finale sopprime la visualizzazione del risultato. I tre puntini (...) indicano che la linea non è finita e ci si aspetta di continuare a capo.

MATLAB

```
1998 100.1 103.2 104.2 104.5 105.9
           102.7 98.4 99.5 98.6;...
1999 96.4 103.5 103.6 104.3 106.6 101.7
           98.5 106.5 98.9;...
2000 102.4 107.6 105.0 106.2 109.4
           102.9 105.1 128.9 107.5;...
2001 107.9 108.9 108.2 108.9 114.6
           104.1 106.4 120.8 108.0];
```

La prima colonna contiene la sequenza degli anni in cui è stata effettuata la misura; mentre ogni colonna descrive un andamento dei prezzi alla produzione dei prodotti industriali di un settore di attività economica. Sempre dalla nostra fonte conosciamo i nomi dei settori economici e decidiamo di usare un altro tipo di "contenitore" per essi. In questo caso la matrice non è adatta a contenere dati che possono avere lunghezza variabile come le stringhe del nostro esempio. Per fare questo è necessario utilizzare un *"cell array"*, cioè array di celle. Qui ogni cella può contenere una qualunque variabile di qualunque dimensione senza nessun riguardo per il fatto che le singole celle debbano avere dimensioni congruenti.

[6]

```
>> categorie = {'Prod. delle miniere e delle cave';...
    'Prod. trasformati e manufatti';...
    'Prod. alimentari, bevande e tabacco';...
    'Prod. dell''industria tessile e dell''abbigliamento';...
    'Cuoio e prod. in cuoio';...
    'Legno e prod. in legno (esclusi i mobili)';...
    'Carta e prod. di carta; stampa ed editoria';...
    'Prod. petroliferi raffinati';...
    'Prod. chimici e fibre sintetiche ed artificiali'};
```

Dobbiamo compiere alcuni ulteriori passi poiché i dati sono ancora in una forma poco maneggevole: il dato dei prezzi è contenuto nella stessa matrice in cui sono contenuti gli anni di riferimento. Ma è semplice creare due matrici distinte. Creiamo il nostro vettore colonna che conterrà gli anni di riferimento:

[7]

```
>> periodo = dati(:,1)
periodo =
    1996
    1997
    1998
    1999
    2000
    2001
```

Facciamo ora qualche cosa di molto simile per estrarre i prezzi dalla matrice dei dati:

[8]

```
>> indiciPrezzi = dati(:,2:end)-100;
indiciPrezzi =
    0.7000  1.8000  2.6000  2.2000  2.7000
            1.2000 -2.8000  5.3000 -1.9000
    6.0000  2.6000  2.9000  3.1000  4.4000
            1.3000 -3.1000  6.5000  0.1000
    0.1000  3.2000  4.2000  4.5000  5.9000
            2.7000 -1.6000 -0.5000 -1.4000
   -3.6000  3.5000  3.6000  4.3000  6.6000
            1.7000 -1.5000  6.5000 -1.1000
    2.4000  7.6000  5.0000  6.2000  9.4000
            2.9000  5.1000 28.9000  7.5000
    7.9000  8.9000  8.2000  8.9000 14.6000
            4.1000  6.4000 20.8000  8.0000
```

Quello che abbiamo fatto qui è stato di prelevare tutte le righe (:) e solo le colonne a partire dalla seconda sino all'ultima. Abbiamo applicato quindi un semplice fattore di scala (100) per rendere i prezzi più facilmente visualizzabili ed apprezzare meglio le differenze. Potremmo ora anche tentare di vedere i nostri dati per mezzo del comando *"plot(periodo,indiciPrezzi)"* ma una visualizzazione così poco raffinata ci informa soltanto sommariamente sul contenuto della matrice. L'idea è quindi quella di produrre una visualizzazione non standard utilizzando un tipo di grafico che ci consenta di apprezzare le peculiarità dei prezzi con più agio. Per questo scegliamo di assegnare un cono di opportuna altezza ad ogni singolo dato. Tratteremo il nostro cono come una superficie formata da tanti spicchi triangolari poggiati su un cerchio di base. Scegliamo di rappresentare il nostro cerchio sul piano per mezzo di un sistema di riferimento polare (in questo sistema i punti appartenenti ad un piano vengono individuati per mezzo di un angolo rispetto ad una retta di riferimento e ad un raggio che misura la distanza dal punto preso come origine). Ogni punto della circonferenza viene descritto per mezzo di due relazioni parametriche: $x = r * cos(theta)$, $y = r * sin(theta)$, dove r è raggio del cerchio e *theta* è l'angolo di cui vogliamo far ruotare r per descrivere una porzione della circonferenza (due *pi greco* radianti se consideriamo l'intera circonferenza). Definiamo quindi il nostro cono come un cilindro degenerato che ha il centro del cerchio di base nell'origine degli assi. Si tratta di un cilindro che possiede una base di raggio r e un cerchio di vertice di raggio pari a zero (un punto). Abbiamo necessità di creare tre matrici (x, y, z) con due righe ognuna. La prima riga descrive il cerchio di base, mentre la seconda quello di vertice. Il numero degli elementi che risiedono in theta é uguale ad $n+1$ poiché vogliamo che il cerchio si chiuda.

[9]

```
>> n = 15 % Numero di punti per i cerchi
n =
  15
>> r = 0.4 % Raggio di base
r =
  0.4000
>> h = 1 % Altezza del cono
```

[6] { }
Le parentesi graffe delimitano la definizione di cell array. Spazi e virgole separano le colonne, mentre i punti e virgola separano le righe, come nelle matrici.

[7] :
I due punti stanno a significare "tutti gli elementi di quella dimensione". In questo caso prendiamo tutte le righe ma soltanto la prima colonna.

[8] End
"end" è una parola riservata che assume automaticamente il valore dell'indice massimo possibile della matrice in oggetto.

[9] %
Il carattere % viene usato per scrivere commenti al codice.

```
h =
    1
[10] >> theta = 2*pi*[0:n]/n % Angoli (n+1) in cui vengono
                                            calcolati i cerchi
theta =
  Columns 1 through 10
         0    0.4189    0.8378    1.2566    1.6755
    2.0944    2.5133    2.9322    3.3510    3.7699
  Columns 11 through 16
    4.1888    4.6077    5.0265    5.4454    5.8643    6.2832
```

Creiamo le coordinate dei due cerchi, di base e di vertice. Ovviamente vorremmo che la seconda riga di x e di y fosse completamente riempita di zeri essendo il cerchio di vertice un punto.
Sappiamo, dalle regole della moltiplicazione tra matrici, che possiamo ottenere questo effetto per mezzo di una moltiplicazione di vettori opportunamente creati:

[11]

```
>> x = [r 0]' * cos(theta)
x =
  Columns 1 through 10
    6.0000    5.4813    4.0148    1.8541   -0.6272
   -3.0000   -4.8541   -5.8689   -5.8689   -4.8541
         0         0         0         0         0
         0         0         0         0         0
  Columns 11 through 16
   -3.0000   -0.6272    1.8541    4.0148    5.4813    6.0000
         0         0         0         0         0         0
>> y = [r 0]' * sin(theta)
y =
  Columns 1 through 10
         0    2.4404    4.4589    5.7063    5.9671    5.1962
    3.5267    1.2475   -1.2475   -3.5267
         0         0         0         0         0         0
         0         0         0         0
  Columns 11 through 16
   -5.1962   -5.9671   -5.7063   -4.4589   -2.4404   -0.0000
         0         0         0         0         0         0
```

All'appello manca una sola variabile: z, cioè l'altezza del cono. È ora intuitivo pensare che debba essere una matrice delle stesse dimensioni di x e y. Essa contiene l'altezza dei punti di base pari a zero (prima riga) e quella dei punti di vertice pari ad h (seconda riga). Per produrre questa matrice scriviamo:

[12]

```
>> z = [0 h]' * ones(1,n+1)
z =
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

Finalmente possediamo il nostro cono (Fig. 2):

```
>> surf(x,y,z)
```

Quello che dobbiamo fare ora è creare un cono per

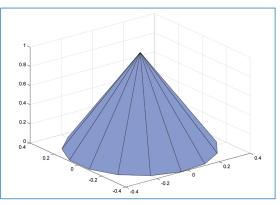



Fig. 2: Il nostro cono visualizzato come tanti triangoli appoggiati al cerchio di base e convergenti in un punto di vertice.

ogni elemento che risiede nella matrice *indiciPrezzi* e colorare in maniera diversa quelli appartenenti a differenti settori in maniera che siano distinguibili dagli altri. Non abbiamo bisogno di ricreare un cono per ogni elemento, sarà sufficiente traslare quello che già possediamo nell'origine e ricalcolare la sua altezza (ricordiamolo, si tratta della seconda riga della matrice z). Una volta che questo è stato fatto, non dovremo fare altro che visualizzare le superfici:

[13]

```
>> [righe,colonne] = size(indiciPrezzi);
>> colori = rand(colonne,3);
>> for w = 1:colonne
    for k = 1:righe
      xx = x + w;
      yy = y + k;
      z(2,:) = ones(1,n+1) * indiciPrezzi(k,w);
      handle = surf(xx,yy,z);
      set(handle,'FaceColor',colori(w,:));
      hold on
    end
  end
```

Nel primo ciclo *"for"*, ad ogni iterazione, la variabile *"w"* assume in sequenza uno dei valori generati dall'espressione a destra dell'uguale (tutti i valori che partono da 1 sino al valore contenuto nella variabile *"colonne"*). I due *"for"* annidati esplorano interamente la matrice *indiciPrezzi* e traslano i valori di x e y che vengono immediatamente utilizzati per la visualizzazione del corrispondente cono. La successiva gestione della parte grafica ha bisogno di una spiegazione un pochino più approfondita. Ogni funzione in grado di generare una visualizzazione può anche restituire quello che viene detto un *"handle"*: un numero che funge da riferimento per l'oggetto grafico appena creato. Questo può essere utilizzato ogni qualvolta vi sia la necessità di variare una proprietà dell'oggetto grafico stesso. E se abbiamo bisogno di cambiare il colore del cono appena generato ecco che il comando *"set(handle,'FaceColor',colori(w,:));"*

MATLAB

[10] [0:n]

Il vettore [0:n] viene moltiplicato per la costante *2*pi/n*, risultando in un nuovo vettore *theta*.

[11] [r 0]'

Notiamo che l'espressione *[r 0]'* produce un vettore colonna.

[12] Ones

La funzione *"ones"* genera una matrice riempita di uno. Le sue dimensioni sono quelle specificate come suoi argomenti.

[13] Rand

La funzione *"rand"* genera una matrice riempita di numeri casuali distribuiti uniformemente nell'intervallo 0,1. Le sue dimensioni sono quelle specificate come suoi argomenti. Qui viene utilizzata per generare i dati RGB che descrivono un insieme di colori.

MATLAB

[14] Gca

La variabile *"gca"* sta per *"get current axis"* e consente di riferirsi all'ultimo asse su cui sono stati visualizzati dei dati.
Se si usa il comando *"get(gca,'Units')"* è possibile vedere quale sia l'unità in uso correntemente.

Sul Web

"Getting Started with MATLAB"
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

"Using MATLAB"
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/using_ml.pdf

Bibliografia

- **FONDAMENTI DI CALCOLO NUMERICO** *Giovanni Monegato* (Edizioni CLUT) 1998
- **METODI NUMERICI E STATISTICI PER LE SCIENZE APPLICATE** *Valeriano Comincioli* (Editrice Ambrosiana) 1992

ci viene in soccorso. Se lo traduciamo letteralmente suona come: *"imposta la proprietà dell'oggetto handle che ha nome FaceColor al valore specificato dalla w-esima riga della matrice colori"*. Normalmente ogni comando di visualizzazione fa scomparire il precedente grafico per lasciare posto al nuovo; la funzione *"hold on"* lo impedisce, facendo in modo che ogni nuovo grafico mostrato sulla figura MATLAB si vada semplicemente ad aggiungere a quelli già esistenti. La trattazione degli oggetti grafici è ovviamente molto più complessa e consente un grande numero di opzioni. Questo argomento lo tratteremo in maggiore dettaglio in un numero futuro quando vi sarà la necessità di sofisticare maggiormente la visualizzazione dei dati, oppure quando dovremo scrivere un'interfaccia grafica. Il risultato finale è visibile in Fig. 3.

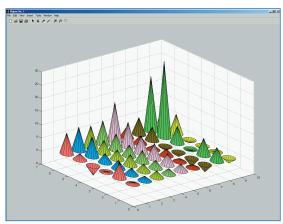
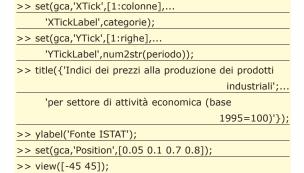



Fig. 3: I coni rappresentano gli indici dei prezzi alla produzione dei prodotti industriali per settore di attività economica.

Operiamo qualche intervento estetico sul nostro grafico. Possiamo notare che manca l'indicazione delle categorie e degli anni sugli assi delle x e delle y. Il seguente codice ci consente di personalizzare il nostro grafico in maniera completa:

[14]

```
>> set(gca,'XTick',[1:colonne],...
        'XTickLabel',categorie);
>> set(gca,'YTick',[1:righe],...
        'YTickLabel',num2str(periodo));
>> title({'Indici dei prezzi alla produzione dei prodotti
                                              industriali';...
        'per settore di attività economica (base
                                              1995=100)'});
>> ylabel('Fonte ISTAT');
>> set(gca,'Position',[0.05 0.1 0.7 0.8]);
>> view([-45 45]);
```

Il risultato finale desiderato è riportato in Fig. 4. Seguendo l'esempio precedente proviamo a descrivere in linguaggio naturale quello che abbiamo fatto con il primo comando:

"imposta la proprietà che regola il numero di marcatori ad un valore pari al numero di colonne e le loro didascalie al valore contenuto nella variabile chiamata 'categorie' sull'asse delle X del grafico corrente".

Ovviamente, il secondo comando esegue la stessa cosa per quanto riguarda l'asse Y, viene poi definito un titolo e viene usata l'etichetta dell'asse delle Y per un'ulteriore informazione descrittiva. Il penultimo comando ci consente di spostare il grafico in una posizione più comoda all'interno della finestra, questo per fare in modo che le didascalie siano più comodamente visualizzate. La proprietà *"Position"* è un vettore di quattro elementi: il primo impone la distanza alla quale si troverà il grafico dal bordo sinistro della finestra, il secondo la distanza dal bordo inferiore della finestra, il terzo la larghezza ed il quarto l'altezza. L'unità di misura è *"Normalized"* e ci consente di lavorare per mezzo di dati percentuali calcolati rispetto alla finestra che ospita il grafico. L'ultimo comando impone un punto di vista definito per mezzo di azimuth ed elevazione (in gradi).

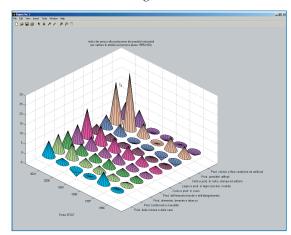



Fig. 4: Dal menu "Tools" selezionare "rotate3D" oppure premere il bottone presente sulla toolbar che riporta una freccia che ruota in senso antiorario, e muovere il grafico con il mouse per ottenere dei punti di vista alternativi.

CONCLUSIONI

In questo articolo abbiamo visto quali siano alcuni dei concetti più comodi per cominciare a lavorare in MATLAB. Abbiamo definito ed utilizzato scalari, vettori e matrici. Inoltre, abbiamo visto come e' possibile fare interagire e manipolare questi oggetti matematici per ottenere i risultati che più ci sono necessari.
Per maggiori informazioni sui prodotti della famiglia MATLAB potete consultare il sito di The MathWorks (*www.mathworks.it*).
Nei prossimi numeri proveremo a utilizzare i primi algoritmi e a comprendere come sia possibile costruire programmi e routine di calcolo numerico.

Fabrizio Sara (fsara@mathworks.it)

Il codice
A BARRE

Tra i sistemi automatici di rilevamento quello basato sul codice a barre è il più diffuso. Scopriamone i segreti e l'implementazione pratica in un linguagio d'alto livello.

Un sistema di rilevamento automatico è costituito da un insieme di elementi Hardware e Software che consentono di acquisire informazioni associate ad oggetti o persone. Attualmente sul mercato sono disponibili tre sistemi di rilevazione automatica: quello basato sul codice a barre (Barcode), quello basato su tecnologia magnetica e quello OCR *(Optical Character Recognition)*. La tecnologia magnetica è utilizzata quando le informazione rilevate devono essere modificate ed archiviate in tempo reale, l'OCR quando le informazioni identificative devono essere fornite anche in chiaro, la tecnologia Barcode, invece, è utilizzata!! … nella maggior parte delle applicazioni commerciali ed industriali, come scopriremo nel corso dell'articolo. Per questo senza dilungarci ulteriormente sui vantaggi e gli svantaggi delle tecnologie, passiamo ad analizzare quella vincente appunto la Barcode. Un sistema di rilevamento basato sul codice a barre, in generale, è costituito da:

1 una simbologia con elementi chiare (spazi) ed elementi scuri (barre) con la quale si identificano gli oggetti delle entità (persone, prodotti ecc.) da rilevare;

2 uno strumento (lettore ottico) che permette di rilevare le barre e convertirle in segnali elettrici;

3 un elaboratore che attraverso delle particolari routine permette di gestire (elaborare, archiviare e stampare) i codici.

I lettori ottici possono essere anche dotati di un sistema di elaborazione e di un display per recuperare informazioni in tempo reale (in questo caso non c'è differenza tra il punto 2 e 3). Nel corso dell'articolo descriveremo i seguenti argomenti:

1 brevemente la storia della simbologia a barre;

2 la simbologia EAN13 e EAN8 (dove EAN sta per *European Article Numbering*);

3 introdurremo un'applicazione Visual Basic che permette di gestire i codici a barre.

4 implementeremo una routine che permette di passare da un codice numerico ad un codice a barre EAN13;

L'applicazione che realizzeremo utilizza un database SqlServer ed un particolare tipo di carattere (EAN13).

IL CODICE A BARRE

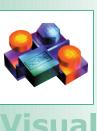
Attualmente abbiamo diverse tipi di codici a barre, per esempio abbiamo codice a barre che rappresentano solo numeri (cioè i codice *EAN13* e *EAN8*), codici che rappresentano tutti i caratteri ASCII (per esempio Code 128) oppure che rappresentano caratteri e lettere (per esempio code39), ecc. In generale un codice a barre è composto da un insieme di barre chiare e scure. Il codice barre di solito contiene informazioni di vario tipo. Per esempio, se il codice identifica un prodotto di un supermercato esso rappresenterà le seguenti informazioni:

- il codice della nazione di provenienza del prodotto,
- il codice del fornitore,
- il codice interno del prodotto
- una cifra di controllo.

Oltre a queste informazioni il codice a barre contiene alcune informazioni di controllo come: barre di start, barre centrali e barre di stop. Il significato di queste informazioni verrà chiarito a breve.

CODICE EAN8 E EAN13

La struttura di questi due codici differisce per il numero di caratteri, che possono essere 8 o 13, e per le informazioni contenute. In particolare nell'EAN8 non è previsto il codice del prodotto.
Allora un codice EAN contiene le seguenti informazioni:

- 2 cifre per indicare il codice internazionale della nazione (per l'Italia è tra 80 e 83)

Visual Basic

Storia del codice a barre

Di codice a barre si inizia parlare già negli anni cinquanta però solo 10 anni dopo, grazie agli sviluppi della tecnologia, vengono prodotti i primi strumenti che li utilizzano. Inizialmente proliferavano i tipi di codifiche, ogni settore ne definiva una. Alla fine degli anni 70 si iniziò a mettere ordine nella babele dei codici con la definizione di alcuni standard. Nasce così: UPC *(Universal Product Code)*, *BAN*, *GenCode*, *EAN* (da cui i due codici che abbiamo descritto in questo articolo). In realtà però non si riuscì a definire uno standard universale, dato che erano già molti e diffusi quelli esistenti.

- 5 cifre per rappresentare il codice dell'azienda, questo codice in Italia è assegnato dall'istituto nazionale per la diffusione del codice dei prodotti (*www.indicod.it*);
- inoltre solo per il codice EAN13 sono previsti 5 cifre per il codice articolo interno all'azienda (di solito si tratta di un progressivo);
- 1 cifra di controllo, detto *Check Digit*, ricavato applicando una formula alle cifre precedenti.

Il *Check Digit* serve a verificare se la lettura del codice a barre da parte dei lettori ottici è stata fatta correttamente. Descriviamo quindi come si calcola il *Check Digit* nel caso dell'EAN13 (per EAN8 la formula è analoga). Si devono considerare, dei 12 caratteri del codice, quelli di posizione pari (2,4,6,8,12) che devono essere moltiplicati per 3 e sommati (nel caso dell'EAN8 i caratteri da considerare sono quelli di posizione dispari). La somma ottenuta va divisa per 10 e il resto va sottratto a 10. Il numero che si ottiene è il tredicesimo carattere, se il numero ottenuto è zero, anche il carattere di controllo sarà zero. In parole povere se il codice di partenza è 801151274230x la somma sarà 33 che diviso 10 porta un resto di 3 quindi un check digit pari a 7. Allora il codice da tradurre in barre sarà: 80 11512 74230 7.

CONVERSIONE IN BARRE

Dopo aver costruito il codice numerico bisogna convertirlo in barre. Ora sommariamente descriveremo di quante barre abbiamo bisogno. Premettiamo, però, che nell'applicazione di esempio utilizzeremo un set di caratteri particolari in cui ad ogni carattere dell'alfabeto corrisponde una rappresentazione in codice EAN13 (all'URL *http://www.microsoft.com/typography/links/links.asp?type=ocr* troverete qualche indicazione). Questo sia perché lo spazio di un articolo è poco e sia perché vogliamo evitare di specificare dettagli sulla codifica.
In EAN13 ogni cifra numerica è rappresentata da 7 barre (o moduli) ed ai moduli delle cifre sono associate i seguenti moduli di controllo:

- i moduli di start (3 moduli)
- i moduli centrale (5 moduli)
- i moduli di stop (3 moduli)

Inoltre bisogna aggiungere che del prefisso internazionale (ricordiamo che è di due cifre numeriche) viene convertita solo la seconda cifra. Quindi se facciamo un po' di calcoli un codice a barra EAN13 è composto da 95 moduli (o barre). Invece l'EAN8 è composto da *95-4x7=67* moduli. Introduciamo un'applicazione di esempio.

L'APPLICAZIONE

L'applicazione, come accennato, utilizza un database SQLServer (di nome CodBarre) con 3 tabelle: *Articoli*, *Codice_barre* e *Progressivo_codice_barre*. Diamo una breve descrizione delle tabelle. La tabella *Articolo* è costituita dai seguenti campi: *Codice_articolo* (interno dell'azienda), *articolo_id* (un progressivo), *Descrizione*. Essa è utilizzata per archiviare le informazioni sugli articoli. La tabella *Codice_barre* è costituita dai seguenti campi: *articolo_id*, *Codice_barre*, *tipo*. In essa verranno salvate le informazioni riguardanti i codici a barre associati ad un determinato articolo. L'unicità del codice a barre viene assicurata da un progressivo archiviato nella tabella *Progressivo_codice_barre*.

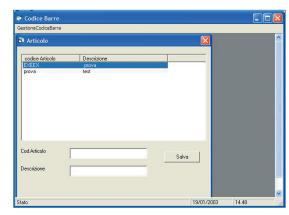



Fig. 1: La form per inserire i codici a barre.

La tabella *Progressivo_codice_barre* è costituita dai seguenti campi: *ProgrEAN8*, *ProgrEAN13* (che sono due progressivi).

IL PROGETTO VISUAL BASIC

Passiamo quindi ad analizzare il codice dell'applicazione. Nel Progetto inseriamo tre form: *frmMain* (una *MDIform*), *frmArticolo* e *frmCodice_barre*. Inoltre inseriamo un modulo, *Modulo1*, di supporto. La *frmMain* la utilizzeremo per richiamare le altre due form, infatti in essa inseriamo un menù a tendina *"GestioneCodiceBarre"* al quale saranno associati rispettivamente due sottomenu *"Articoli"* e *"Codice barre"*. Il codice presente sulla *frmMain* gestisce essenzialmente la creazione di nuove istanze delle due *form*. Dovete prevedere delle istruzioni come le seguenti:

```
Dim f As New frmCodice_barre
f.Show
```

La *frmArticolo* è una semplice form che utilizziamo per gestire gli articoli, contiene una *ListView*, due textbox e un pulsante. La *Listview* serve a mostrare gli articoli mentre i due *Textbox* e il pulsante servono per inserire un nuovo articolo nel database.
Di questa form descriviamo solo il codice per la lettura dei dati degli articoli. Notate che per visualizzare gli articoli sulla *Listview1* utilizziamo il Recordset *rs*, ecco il codice:

```
Private Sub Form_Load()
 Dim rs As ADODB.Recordset
 Dim itmx As MSComctlLib.ListItem
 Set rs = New ADODB.Recordset
 rs.Open "select * from articolo", connAdo
 Do Until rs.EOF
    For i = 0 To rs.Fields.Count - 1
      Set itmx = Me.ListView1.ListItems.Add(, ,
                                        CStr(rs(i).Value))
      i = i + 2
      itmx.SubItems(1) = CStr(rs(i).Value)
    Next
     rs.MoveNext
Loop
```

La connessione ADO è definita nel modulo con la seguente dichiarazione:

```
Public connAdo As ADODB.Connection
```

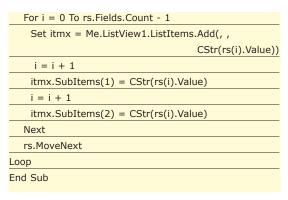
La connessione, sempre nel modulo, può essere generata tramite una procedura simile alla *Main()* descritta in seguito. In essa abbiamo previsto anche la *Show* della *frmMain*.

```
Public Sub Main()
 Set connAdo = New ADODB.Connection
    With connAdo
      .ConnectionTimeout = 1000
      .Open "codbarre", "sa", miaPassword
    End With

    Set fMainForm = New frmMain
    fMainForm.Show
End Sub
```

Descriviamo il nucleo centrale dell'applicazione cioè la form *frmCodice_barre* e le procedure per la generazione del codice a barre. Sulla *frmCodice_barre* posizioniamo due *Listview* che ci serviranno per selezionare l'articolo di cui vogliamo creare il codice e per la visualizzazione del codice generato.
Inseriamo anche: due textbox per la visualizzazione del codice a barre in formato numerico e grafico; un combobox per selezionare il tipo di codice da creare e tre pulsanti per generazione, salvataggio e cancellazione del codice generato. Per visualizzare le informazioni riguardanti l'articolo utilizziamo una procedura come la seguente.

```
Private Sub VisualizzaArticoli()
Dim rs As New ADODB.Recordset
Dim str As String
Dim itmx As MSComctlLib.ListItem
 str = " select articolo.codice_articolo,articolo.descrizione,
                                        articolo.articolo_id"
 str = str + " from  articolo "
 rs.Open str, connAdo
 Do Until rs.EOF
```

```
    For i = 0 To rs.Fields.Count - 1
      Set itmx = Me.ListView1.ListItems.Add(, ,
                                        CStr(rs(i).Value))
       i = i + 1
      itmx.SubItems(1) = CStr(rs(i).Value)
      i = i + 1
      itmx.SubItems(2) = CStr(rs(i).Value)
    Next
    rs.MoveNext
Loop
End Sub
```

Per generare il codice a barre effettueremo un doppio passaggio: prima "formatteremo" il codice numerico secondo le nostre esigenze e poi, attraverso la funzione di conversione *CarattereEAN13* (che ripetiamo utilizza i caratteri EAN13), trasformeremo il codice numerico in un codice a barre.

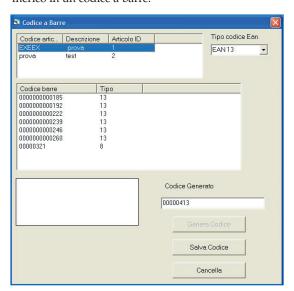
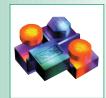



Fig. 2: **La form per gestire i codici a barre.**

Successivamente il codice a barre, con qualche accorgimento, potrà essere stampato per esempio attraverso un *Report Object*.

FORMATTAZIONE E CONVERSIONE

La formattazione, nel nostro caso, consiste nel: generare un progressivo, a partire dal numero archiviato nella tabella *Progressivo_codice_barre*; creare una stringa di 12 caratteri con degli zeri sulla sinistra; calcolare il *Check digit* e aggiungerlo alla stringa di 12 caratteri così da ottenere il codice completo. Riassumendo, la stringa formattata contiene: il check digit, il progressivo e tanti zeri quando sono le cifre mancanti per raggiungere 13. Questa stringa, infine, verrà passata alla funzione di conversione e il risultato mostrato su un Textbox. La visualizzazione del codice a barre è fatta sul textbox *text1* (come mostrato in Fig. 2) al quale, alla proprietà font abbiamo assegnato il tipo EAN13. Facciamo notare, che con tutti questi passaggi dal no-

Significato delle singole barre e dei numeri?

Le prime due cifre si riferiscono al Paese: per esempio i numeri 80-83 stanno per Italia, 90-91 per Austria e così via. Questi numeri non dicono però quale sia l'origine del prodotto o la provenienza delle materie prime; le successive 5 cifre rappresentano l'indirizzo del produttore oppure del fornitore; e le 5 cifre ulteriori si riferiscono all'articolo stesso. Per esempio possono significare: cioccolatini assortiti, 100 g, confezione regalo ecc ...; l'ultimo numero serve solamente come verifica, in modo che il computer possa accorgersi di un'eventuale "svista".

Visual Basic

Tecnologia a barre

Il successo del sistema codice a barre è stato decretato da vari fattori: dalla simbologia binaria (barre di due tipi) vicina al mondo del computer; dalla efficienza (affidabilità, velocità, economicità) dei sistemi di riconoscimento e di stampa (non sono necessarie stampanti particolari) ed in generale dalla flessibilità della tecnologia che si adatta ad ogni settore operativo.

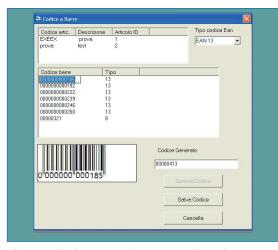



Fig. 3: Nella form mostriamo un codice a barre generato con la nostra funzione.

stro codice a barre, omettiamo la parte di codice (5 caratteri) che costituiscono il codice azienda, infatti, lo poniamo tutto a zero. Per eseguire questi passaggi possiamo usare delle funzioni simili a quelle che descriveremo tra poco. L'avvio di queste funzioni viene fatta attraverso il pulsante *Genera_codice* Nella procedura precedente se il codice da generare è EAN13: formattiamo la stringa generiamo il codice a barre e incrementiamo il progressivo (per la successiva generazione). Il progressivo viene incrementato direttamente sul database tramite un'operazione di *Update*. Notate che la funzione ricerca progressivo esegue la lettura del progressivo dalla tabella *progressivo_codice_barre*. In *\soft\codice\barcode\codicivari.txt* trovate diverse funzioni dell'applicazione.

VISUALIZZIAMO IL CODICE A BARRE

Finalmente ora ci occuperemo della visualizzazione del codice a barre. Innanzitutto mostriamo le procedure che permettono di caricare le listview. Ecco come si carica la listview che mostra i codici a barre. È Necessario inserire il codice per caricare la listview1 con l'articolo.

```
Private Sub ListView1_ItemClick(ByVal Item As
                              MSComctlLib.ListItem)
If Item.Text = "" Then Exit Sub
    ListView2.ListItems.Clear
    codListView = Trim(Item.SubItems(2))
    AggiornaListview2 codListView
End Sub
```

Cliccando su una riga della *listview1* si aggiorna la *lisview2*.

```
Public Sub AggiornaListview2(Codice As String)
Dim itmx As MSComctlLib.ListItem
Dim rs As New ADODB.Recordset
Dim str As String
str = "select codice_barre.codice_barre,
                              codice_barre.tipo_codice"
 str = str + " from codice_barre , articolo "
 str = str + " where codice_barre.articolo_id=
           articolo.articolo_id and articolo.articolo_id=" & ""
                                          & Codice & ""
 rs.Open str, connAdo
 Do Until rs.EOF
    For i = 0 To rs.Fields.Count - 1
      Set itmx = Me.ListView2.ListItems.Add(
                                  , , CStr(rs(i).Value))
      i = i + 1
      itmx.SubItems(1) = CStr(rs(i).Value)
 Next
    rs.MoveNext
Loop
End Sub
Private Sub ListView2_ItemClick(ByVal Item As
                              MSComctlLib.ListItem)
If Item.Text = "" Then Exit Sub
   Text1.Text = VisualizzaCodiceBarre(Trim(Item.Text))
End Sub
```

Quando si clicca sulla *listview2* viene mostrato il codice a barre. La funzione che trasforma il codice in barre è la *CarattereEAN13*

```
Public Function VisualizzaCodiceBarre(testo As String)
                                          As String
   VisualizzaCodiceBarre = CarattereEAN13(Mid(testo, 1,
                                       Len(testo) - 1))
End Function
```

LA PROCEDURA DI CONVERSIONE

Come accennato, la procedura CarattereEAN13, dal codice numerico ricava il codice a barre. E' implementata in base a un set di caratteri EAN13, prevede i seguenti elementi.

- Una matrice 3x10, che contiene i caratteri dell'alfabeto in minuscolo e i caratteri da A a D in maiuscolo.
- Un array di 10 cifre "binarie" che servirà per selezionare gli elementi della matrice di caratteri. La funzione è descritta nel file *\soft\codice\barcode \carattereEAN13.txt*

CONCLUSIONE

In questo appuntamento, sommariamente, abbiamo fatto "la storia" del codice a barre ed abbiamo introdotto un'applicazione Visual Basic che li gestisce. In conclusione vi invitiamo a completare l'applicazione introdotta ed a convertire i codici a barre senza utilizzare i caratteri EAN13 (non è semplice! …).

Massimo Autiero

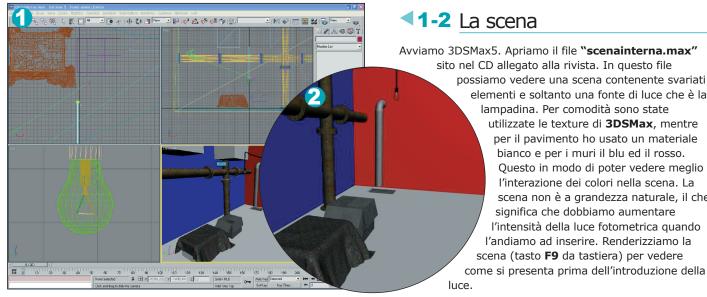
Illuminazione avanzata

3D Studio Max

Pietro Canini

Scopriamo i nuovi motori di Global Illumination e Radiosity integrati in 3D Studio Max 5. Il programma ci mette a disposizioni due possibilità: il Light Tracer, ed il Radiosity; andiamo a vedere come funzionano.

La Discreet dopo tanto tempo integra in 3D Studio Max 5 un nuovo motore di rendering capace di renderizzare le scene in modo realistico, generando quindi un'illuminazione che interagisce con tutto l'ambiente. Abbiamo quindi due soluzioni: l'opzione *Light Tracer* ed il *Radiosity*. Il *Light Tracer*, se attivato, genera delle ombre con bordi morbidi e delle sfumature di colore tra esse e gli oggetti presenti nella scena. Al contrario del *Radiosity*, il *Light Tracer* non tiene conto delle simulazioni fisicamente corrette della scena, quindi uno dei suoi pregi è la velocità. Da questi fatti si può dedurre che questa tecnica di rendering si utilizza spesso in scene esterne. Non conviene usarla in scene di interni, in quanto questo compito è lasciato al *radiosity*. La tecnica *radiosity* di 3DSMax5 produce delle simulazioni fotometriche accurate delle luci presenti nella scena. Attivando questa opzione quindi avremo nei nostri rendering effetti come la luce indiretta, ombre morbide, e colori sfumati. In congiunzione con queste tecniche, c'è l'introduzione di nuove tipologie di luci (oltre alle vecchie presenti) che è possibile settare attraverso le unità fotometriche reali, e non tramite valori arbitrari. Per usare il *Radiosity* in scene fisicamente realistiche è conveniente: utilizzare per la scena, quando possibile, dimensioni reali utilizzando ad esempio unità come il millimetro (mm.), centimetro (cm.) e così via; lavorare esclusivamente con luce fotometriche reali in modo di avere sotto controllo l'intensità della luce nella scena; per simulare invece la luce naturale, tipo il sole, utilizzare le luci *IES Sun* e *IES Sky*; essere sicuri che il materiale usato nella scena abbia un valore di riflettanza coerente; attivare il *Controllo di esposizione* per avere i risultati finali desiderati. Per avere il *Radiosity* non è necessario usare luci fotometriche e materiali complessi, bisogna però tener conto di alcuni fattori: se utilizziamo le luci standard di 3DSMax5, il motore di rendering radiosity interpreta queste luci come fossero le fotometriche, quindi, ad esempio, se utilizzo una luce spot standard con intensità (multiplier value) ad *1.0* l'algoritmo capirà che quella è una luce fotometrica spot con un'intensità di 1500 candelas (valore di default); per la luce naturale, cioè il sole, oltre a quanto descritto sopra, si può soltanto usare una luce diretta per il sole, ed uno *Skylight* per avere l'illuminazione diffusa prodotta dal sole; attivare il controllo di esposizione su logaritmico (*Logarithmic Exposure Control*) e spuntare l'opzione *Affect Indirect Only*. Se vogliamo il *Radiosity* anche in animazione, basta spuntare l'opzione *Compute Advanced Lighting When Required* nella finestra di rendering. Per capire quanto fin ora detto iniziamo con la pratica...

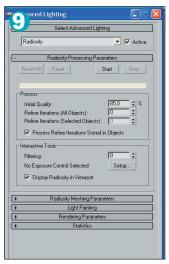
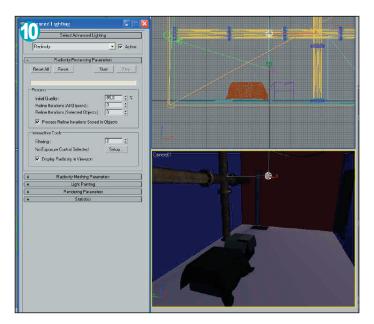
1-2 La scena

Avviamo 3DSMax5. Apriamo il file **"scenainterna.max"** sito nel CD allegato alla rivista. In questo file possiamo vedere una scena contenente svariati elementi e soltanto una fonte di luce che è la lampadina. Per comodità sono state utilizzate le texture di **3DSMax**, mentre per il pavimento ho usato un materiale bianco e per i muri il blu ed il rosso. Questo in modo di poter vedere meglio l'interazione dei colori nella scena. La scena non è a grandezza naturale, il che significa che dobbiamo aumentare l'intensità della luce fotometrica quando l'andiamo ad inserire. Renderizziamo la scena (tasto **F9** da tastiera) per vedere come si presenta prima dell'introduzione della luce.

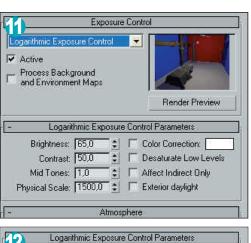
3-4-5 Aggiunta della luce fotometrica

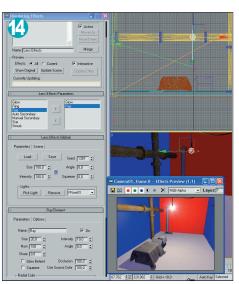
Nella vista **Left** facciamo uno zoom sulla lampadina (se già non è fatto). Apriamo il pannello **Create/ Lights / Photometrics** e clicchiamo su **Free Point**. A questo punto, nella vista **Left** creiamo la luce fotometrica e posizioniamola proprio al centro della lampadina, aiutandoci anche con la vista **Top** (Figura 05). Con la luce appena creata selezionata, apriamo il pannello **Modify** per definirne i parametri.

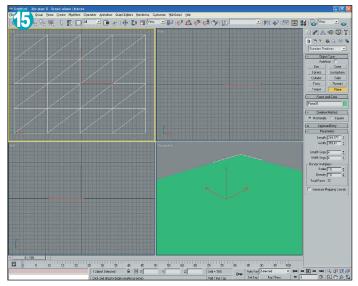
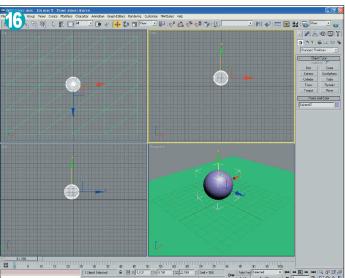
6-7-8 I parametri

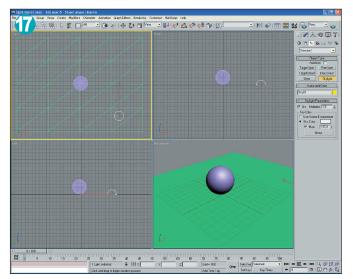
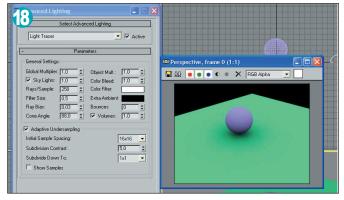
Le ombre lasciate attivate le **Shadow Map** di default. Poi, a seguire, clicchiamo sul tasto **Exclude**, selezioniamo dalla lista l'oggetto **Object** (che sarebbe il vetro della lampadina) e portiamolo alla parte destra (con le due freccette a destra) per escluderlo dalla luce. Questo perché l'ombra **Shadow Map** non tiene conto dell'opacità del vetro quindi, stando la luce fotometrica all'interno del vetro, genera un'ombra nera che praticamente non fa vedere niente. Sotto **Color** impostiamo **Incandescent** (per simulare la luce della lampadina al tungsteno) e come intensità mettiamo il valore 100000. Ora provate ad eseguire il rendering e vedete l'effetto della luce (se la ritenete ancora troppo bassa o alta, variate di conseguenza l'intensità).

Calcolo del Radiosity 9-10

Apriamo il pannello **Advanced Light** (tasto **9** da tastiera). Dalla lista scegliamo **Radiosity**. Nella finestra che ci appare lasciamo i parametri di default, ma sappiate che i valori **Initial Quality** e **Refine Iterations** servono ad aumentare la qualità dell'immagine finale, a discapito dei tempi di calcolo. In **Filtering** impostiamo il valore **2**; più questo numero è grande, più i disturbi di colore (diciamo lo sporco), generato dal **radiosity**, viene attenuato. A questo punto clicchiamo su **Start** per far partire il calcolo. Aspettiamo finché non arriva ad **85%** (valore settato in **Initial Quality**).
Dato che l'opzione **Display Radiosity** in **Viewport** è attivata, il **Radiosity** generato verrà visualizzato nella vista **Camera01** e possiamo muoverci all'interno per apprezzare i risultati ottenuti. Tra gli altri parametri del **Radiosity** troviamo: **Global Subdivision Settings** permette di determinare se vogliamo una mesh o no, e impostare la grandezza degli elementi della **mesh**, questo perché 3DSMax calcola l'intensità tramite punti discreti nell'ambiente suddividendo le **mesh** originali in elementi che fanno parte del **radiosity**; il **Light Painting** ci permette di dipingere ombre o illuminare le superfici in modo di rimuovere gli artefatti generati dal **radiosity** senza riprocessare il tutto di nuovo; l'ultimo **rullout** ci permette di controllare in che modo sia processato il **radiosity**.

11-12 I controlli di esposizione

Non è tutto, bisogna attivare ancora il controllo di esposizione. In **Interactive Tools** clicchiamo sul tasto **Setup**. Ci verrà aperta la finestra **Environment** nella quale possiamo attivare i parametri di esposizione. Nel **rullout Exposure Control** scegliamo nel menù a discesa l'opzione **Logarithmic Exposure Control** (di solito è la scelta migliore) ed effettuiamo un rendering per vedere i risultati. La scena risulta troppo illuminata, mettiamo quindi il valore **Brightness = 55** e **Contrast = 55** (di solito i valori di default vanno più che bene). **Mid Tones** lasciamolo ad **1** e **Physical Scale** in questo caso non serve perché viene utilizzato quando in scena ci sono luci non fotometriche (quelle standard) quindi lasciamolo a 1500. Renderizzate e vedete la scena col **radiosity**.

13-14 Aggiunta del bagliore e rendering finale

Manca qualcosa, cioè il bagliore intorno alla lampadina. Aprite il pannello **Effects** (menù a discesa **Rendering /Effects...**) clicchiamo sul tasto **Add...** e dalla lista scegliamo **Lens Effects**. In **Lens Effects Parameters** portiamo a destra la scritta **Glow** e **Ray**. Sotto **Lights** clicchiamo sul tasto **Pick Light** e selezioniamo la nostra luce fotometrica nella scena. Con la scritta **Ray** selezionata (quella nella parte destra) spostiamoci nei suoi parametri **Ray Elements** ed impostiamo **Size = 20** dato che col valore **300** erano troppo grandi. Ora renderizzate e la scena è finita.

15-16 Il Light Tracer

Resettate la scena (**File / Reset**). Iniziamo a creare una scena esterna per vedere come funziona il **Light Tracer.** Aprite il pannello **Create / Geometry / Plane** e create nella vista **Top** un piano. Ora creiamo una sfera e posizioniamola sopra al piano. Ecco creata la nostra scena esterna, serve poco per apprezzarne subito il funzionamento.

Il sole e la scena finale 17-18

Aggiungiamo ora una sorgente di luce, questa volta non fotometrica. Apriamo quindi il pannello **Create / Lights / Skylight** e nella vista **Top** clicchiamo per creare il sole. È indifferente la posizione di questa sorgente luminosa in quanto prende in considerazione tutta la scena. Apriamo il pannello **Advanced Light** (tasto **9** da tastiera) e dal menù a tendina scegliamo **Light Tracer**. A questo punto lasciando i valori di default già possiamo vedere i risultati ottenuti facendo il rendering. Al contrario del **Radiosity**, il **Light Tracer** non è visualizzabile nella **Viewport** ed esegue il calcolo ogni volta che viene lanciato il rendering. È possibile aggiungere anche una normale luce spot che genera ombra, per raffinare il gioco di ombre che abbiamo nel rendering.

Veloce panoramica 19

Vediamo velocemente a che servono i parametri. Il **Global Multiplier** controlla il livello di luce nella scena; **Object Mult.** controlla il livello di luce riflessa dagli oggetti; **Sky Light** regola l'intensità della luce del cielo; **Color Bleed** controlla la forza dei colori nella scena sugli altri; **Rays/Sample** equivale al numero di raggi generati dagli oggetti (le classiche macchie che si formano nella scena con questo valore basso); **Color Filter** filtra il colore della luce con quello selezionato; **Filter Size** corrisponde alla grandezza in pixel del filtro usato per ridurre il disturbo generato dal **Ray / Sample** con valore basso; **Extra Ambient** aggiunge un altro colore (quello ambientale) alla scena; **Ray Bias** aggiusta la posizione della luce rimbalzata; **Bounces** equivale al numero di raggi che vengono rimbalzati da un oggetto all'altro; **Cone Angle** controlla l'angolo usato per la prima partitura di luce; **Volumes** ha effetto quando vengono utilizzati le luci oppure la nebbia volumetrica. In figura si può vedere un'immagine creata tramite il corretto utilizzo di questi parametri.

20-21 Light Tracer con luce fotometrica

Nella stessa scena cancelliamo la sorgente di luce creata poco fa, rechiamoci quindi nel pannello **Create / Lights / Photometrics** e creiamo uno **IES Sun** con ombra attivata (**Shadows** su **ON**) ed uno **IES Sky** con nella vista **Front**.
Aprite la finestra **Environment** (tasto **8** da tastiera), sotto **Exposure Control** selezionate **Logarithmic Exposure Control** e nei suoi parametri attiva **Exterior Daylight** ed il gioco è fatto.
Provate a renderizzare e ammirate i risultati ottenuti.

File sul CD
\3D Studio Max

Conclusioni

In conclusione abbiamo visto teoricamente e praticamente come funzionano questi nuovi motori di rendering e la velocità con cui essi vengono eseguiti.
Per maggiori informazioni fate sempre riferimento alla guida di 3DStudio e sperimentate sempre.
Utilizzate il minor numero di sorgenti luminose possibili perché queste aumentano di molto i tempi di rendering.
Per gli oggetti che non si vedono, o vogliamo non vengano processati dal **radiosity**, basta cliccare col tasto destro su di essi e scegliere **Properties**, andare nella finestra **Adv. Lighting** ed escludere quello che vogliamo (luci, ombre, ecc…).
Nel **Material Editor** invece abbiamo il nuovo materiale **Advanced Lighting Override** da utilizzare soprattutto quando abbiamo oggetti auto-illuminati, fosforescenti ecc…
Trovate le scene finali nel CD-Rom allegato alla rivista.
Con questo è tutto ed a presto col prossimo articolo.

Testare
IL CODICE CON JUNIT
PARTE SECONDA

Junit

Il mese scorso abbiamo scoperto gli "unit test" e abbiamo visto come possiamo testare il codice e divertirci allo stesso tempo. In questo secondo articolo della nostra breve serie impareremo qualcosa di nuovo su questa pratica straordinaria, alla ricerca di un codice senza errori e di un obiettivo quasi metafisico: il ritmo.

Nell'articolo del mese scorso abbiamo provato per la prima volta le librerie JUnit. Prima abbiamo scritto una semplice classe Java che rappresenta un movimento di denaro su un conto corrente. Poi abbiamo scritto un test unitario per verificare che la classe funzioni bene. Ecco la classe *Importo*:

```
package serie_junit;

/**
 *  Un importo in denaro.
 */
public class Importo
{
  private final boolean _positivo;
  private final long _euro;
  private final long _cent;

  /**
   *  Costruttore per l'oggetto Importo
   *
   *@param  positivo  True se l'importo è in ingresso,
                                  false se è in uscita.
   *@param  euro      Importo in Euro (esclusi
                         centesimi) - sempre positivo.
   *@param  cent      Centesimi - sempre positivo.
   */
  public Importo(boolean positivo, long euro, long cent)
  {
     _positivo = positivo;
     _euro = euro + (cent / 100);
     _cent = cent % 100;
  }

  /**
   *  Restituisce l'importo come una stringa.
   *
   * @return    Il valore dell'importo.
   */
  public String getValore()
  {
     String risultato = "";
     if(!_positivo)
       risultato += "-";
       risultato = risultato + _euro + ".";
     if(_cent < 10)
       risultato += "0";
     risultato += _cent;

     return risultato;
   }
}
```

Ed ecco il test unitario per la classe Importo:

```
package serie_junit;
import junit.framework.*;

/**
 *  Test case per la classe Importo.
 */
public class TestImporto extends TestCase
{
  public static void main(String[] args)
  {
     junit.swingui.TestRunner.run(TestImporto.class);
  }

  /**
   *  Test per il metodo getValore().
   */
  public void testGetValore()
  {
    Importo i1 = new Importo(true, 12, 06);
    assertEquals(i1.getValore(), "12.06");
    Importo i2 = new Importo(false, 0, 99);
    assertEquals(i2.getValore(), "-0.99");
```

File sul CD
\soft\codice\junit3.8.1.zip

Sul CD allegato alla rivista troverete la libreria JUnit e tutto il codice sorgente di questo articolo.

Junit

```
    Importo i3 = new Importo(true, 12, 100);
    assertEquals(i3.getValore(), "13.00");
    Importo i4 = new Importo(false, 12, 375);
    assertEquals(i4.getValore(), "-15.75");
  }
}
```

Sembra che per ora vada tutto bene. Ma se vogliamo essere pignoli esiste ancora una possibilità di errore, alla quale avevamo già accennato. Quando costruiamo un *Importo*, i parametri numerici che rappresentano gli Euro e i centesimi devono avere sempre un valore positivo (il fatto che l'importo sia "in entrata" o "in uscita" viene deciso dal campo *_positivo*). Ma cosa succede se qualcuno cerca di costruire un Importo con valori numerici negativi?

```
Importo i = new Importo(true, -10, -2);
```

Cosa succederebbe in un caso come questo? Ad esempio, cosa otterremo in uscita dal metodo *getValore()*?

Fig. 1: Il sito ufficiale di Junit si rivela una vera miniera di informazioni. www.junit.org.

Non lo sappiamo. Non abbiamo previsto che il client usi valori negativi, e quindi non sappiamo cosa farebbe il codice. Potremmo cercare di gestire tutti i vari casi esplicitamente, ma è inutile complicarci la vita. È meglio proibire semplicemente al client della classe di usare valori numerici negativi nel costruttore. Dopo tutto esiste già un parametro apposta per definire un importo negativo. Quindi se qualcuno sta cercando di usare dei numeri negativi nel costruttore allora sta probabilmente sbagliando, ed è il caso di segnalarglielo.

VOSTRO ONORE, HO UN'ECCEZIONE

Possiamo usare il meccanismo delle eccezioni per risolvere questo problema. Se qualcuno cerca di costruire un Importo con dei numeri negativi, il costruttore deve restituire un'eccezione. Potremmo creare una nuova eccezione apposta, ma Java ha già una *NumberFormatException* che fa proprio al caso nostro: ha un nome espressivo ed è una *RuntimeException*, quindi non dobbiamo scomodarci a controllarla con un blocco *try-catch* ogni volta che costruiamo un *Importo*. Decidiamo quindi che il costruttore di *Importo* deve lanciare una *NumberFormatException* se i due parametri numerici non sono entrambi positivi.
Naturalmente dovremo scrivere un test unitario per questo codice, cioè un nuovo metodo per la classe *TestImporto*. Dopo tutto abbiamo deciso che d'ora in poi scriveremo solo codice testato. Prima di scrivere il codice, però, fermiamoci un momento e proviamo a fare un gioco. Visto che comunque dobbiamo scrivere il test, vi proponiamo di procedere al contrario: prima scriviamo il test e poi scriviamo il codice. La cosa può sembrare strana, ma tra poco vi spiegheremo il perché di questa idea balzana.
Occupiamoci del test, dunque. Come facciamo a verificare che il costruttore lanci l'eccezione? Il nostro test può provare a "sbagliare" volutamente, e poi verificare se salta fuori l'eccezione che ci aspettavamo:

```
public void testCostruzioneSbagliata()
{
  try
  {
    new Importo(true, -1, 0);
    fail("Mi aspettavo un'eccezione");
  }
  catch (NumberFormatException e) {}

 try
 {
   new Importo(true, 0, -1);
   fail("Mi aspettavo un'eccezione");
 }
 catch (NumberFormatException e) {}
}
```

Il test prova a costruire un *Importo* con dei valori illegali all'interno di un blocco *try*. Osservate la logica, perché questo è un ragionamento che possiamo applicare ogni volta che vogliamo testare le eccezioni: se l'operazione genera una *NumberFormatException*, come ci aspettiamo, il flusso del programma "salta" al blocco *catch*, che non fa niente, e l'esecuzione prosegue. Se invece l'operazione non genera una *NumberFormatException*, allora si passa semplicemente all'istruzione successiva. Ma l'istruzione successiva è una chiamata al metodo *fail()* della classe *TestCase*, che ordina al test di fallire immediatamente (eventualmente con un messaggio esplicativo, come in questo caso).
Ecco lo schema:

Cosa dovremmo testare?

Dovremmo testare tutto. O meglio: tutto quello che potrebbe non funzionare. Ogni singolo metodo non private di ogni singola classe dovrebbe avere il suo test.

```
try
{
  // la seguente operazione dovrebbe generare
  // un'eccezione:
  faiQualcosaDiSbagliato();
  // se siamo arrivati qui, vuol dire che non
  // c'e' stata nessuna eccezione. Il test deve
  // fallire!
  fail();
}
catch (Exception e)
{
  // se abbiamo avuto un'eccezione, l'esecuzione
  // del codice continua da qui
}
```

Dato che vogliamo testare il codice con attenzione quasi maniacale, abbiamo ripetuto questa stessa logica due volte per i due parametri numerici del costruttore. Ora proviamo a fare girare l'intero "test case" (come al solito basta eseguire il *main()* della classe *TestImporto*). Il risultato è una barra rossa e il messaggio "Mi aspettavo un'eccezione". Ovviamente il test è fallito, perché non abbiamo ancora scritto il codice che gli permette di passare.
Dopo aver scritto il test passiamo dunque alla seconda parte del nostro gioco: scriviamo il codice. Modifichiamo il costruttore della classe *Importo* perché generi l'eccezione che vogliamo:

```
public Importo(boolean positivo, long euro, int cent)
{
  _positivo = positivo;
  if(euro < 0 || cent < 0)
    throw new NumberFormatException("Valori negativi
                              nel costruttore di un Importo");
  _euro = euro + (long)(cent / 100);
  _cent = cent % 100;
}
```

Non c'è bisogno di dichiarare l'eccezione con la parola chiave *throws*, perché si tratta di una *RuntimeException*. Facciamo girare di nuovo il nostro test, e riecco la sospirata barra verde!

UNA QUESTIONE DI RITMO

Proviamo a riflettere su quello che abbiamo appena fatto. Stiamo continuando a testare tutto il codice che scriviamo, ma questa volta abbiamo invertito le due operazioni: prima abbiamo scritto il test, e poi il codice. Perché questa strana scelta? Se vogliamo scrivere codice davvero solido, allora non dovremmo considerarlo completo fino a quando non abbiamo finito di scrivere i test. I test ci tranquillizzano sul fatto che il codice funziona, ma allo stesso tempo documentano il modo in cui il codice dovrebbe funzionare. Insomma, non sono solo una verifica, ma anche una specifica. Quindi tanto vale scriverli prima dello stesso codice. In questo modo potremo subito renderci conto di come funzionerà il codice quando lo avremo scritto.
Dopo aver scritto il test, scrivere il codice diventerà un'operazione abbastanza semplice e meccanica. L'obiettivo è sempre quello di scrivere meno codice possibile, giusto quel tanto che basta per far girare il test. Quando il test gira, allora abbiamo finito di implementare una funzionalità e possiamo passare alla successiva. Nel gergo dei programmatori che l'hanno inventata, questa tecnica si chiama *"Test-First Design"*.
Ormai stiamo cominciando a capire la filosofia dei test unitari. Stiamo alternandoci in continuazione tra il codice e i test, scrivendo il nostro codice a piccoli pezzi. Pezzo su pezzo possiamo costruire un programma enorme, con tutti i suoi bravi test. Questa operazione richiede una certa sfuggente qualità che possiamo chiamare "ritmo": ha ritmo chi si concentra su una singola funzionalità senza lasciarsi distrarre da modifiche estemporanee, alterna regolarmente la scrittura dei test a quella del codice, non lascia mai codice senza test e fa girare i test in continuazione. Il ritmo è una capacità acquisita ma, se impariamo a farla nostra, trasforma la programmazione in un'attività molto più semplice e piacevole.
Niente più notti insonni di debugging dopo poche ore di furiosa e disordinata crescita del codice: i nostri programmi crescono lentamente ma costantemente, i bug emergono quasi sempre subito, e quasi sempre sappiamo subito dove andarli a cercare.

MA NE VALE LA PENA?

Fino ad ora abbiamo testato tutto il codice che

Quando dovremmo testare?

Dovremmo testare in continuazione, alternando la scrittura dei test a quella del codice. Non dovrebbe quasi esistere una distinzione tra le due attività.
Ciascun test andrebbe scritto non dopo aver scritto il codice, ma prima (Test-First Design), come spieghiamo in questo stesso articolo. Il codice è completo quando il test dà un responso positivo.
Questo significa che un frammento di codice non testato deve essere per principio considerato incompleto, o meglio ancora sbagliato.

Junit

Diffusione

Il crescente successo del pattern proposto da Junit si rispecchia in una sempre maggiore diffusione di ambienti di sviluppo che lo supportino: JBuilder, Eclipse, Forte, Visual Age, IntelliJ, TogheterJ, Kawa, solo per citare i più noti. Questi ambienti visuali consentono anche agli utenti meno smaliziati di godere dei benefici della programmazione guidata dagli "unit test".

Junit

Cinque scuse per non testare il software

Dì la verità, figliuolo: hai testato regolarmente il tuo codice negli ultimi mesi? Ah, no? E per quale motivo? Scommettiamo che la tua risposta sarà tra le seguenti.

"NON HO ABBASTANZA TEMPO" – Questa è sempre la scusa numero uno. C'è sempre qualcosa di urgente da fare, più urgente dei test - tipicamente correggere qualche bug o implementare qualche nuova caratteristica. Ma il tempo che perdi oggi scrivendo test (attività niente affatto spiacevole) sarà tempo guadagnato con gli interessi domani, quando non sarai costretto a passare ore curvo su un debugger (attività sgradevolissima e frustrante).

"MI ANNOIO" – La gran parte dei programmatori pensa che scrivere test, controllarne il risultato, eccetera, sia meno divertente e stimolante che scrivere "vero codice". Ma la tua noia svanirà come neve al sole quando capirai che scrivendo un po' di test verrai gratificato da una bella barra verde e dalla consapevolezza che il tuo codice funziona.

"SO GIÀ CHE FUNZIONA" – I test servono a scoprire se qualcosa nel codice è sbagliato. Ma la gran parte dei problemi possono essere scoperti semplicemente facendo girare il codice per un po', vero? Guarda in faccia la realtà: non è mai stato così, inutile ostinarsi a negarlo. Tu non sai che il codice funziona fino a quando i test non girano. E soprattutto, il codice che funziona oggi potrebbe non funzionare più domani, quando lo avrai modificato.

"CI PENSERÀ QUALCUN ALTRO" – Nelle grandi aziende esistono dipendenti pagati apposta per testare il codice. Quindi qualcun altro può testare il sistema come una "scatola nera". Ma solo tu che lo stai scrivendo puoi testare il codice a basso livello e individuare la maggior parte dei problemi prima che si manifestino.

"I TEST SONO INUTILI" – È impossibile scovare ogni possibile bug con i test, e i programmatori sono tipicamente poco bravi a capire dove potrebbero annidarsi i bug. Quindi spesso si finisce per testare proprio quello che funziona anziché quello che non funziona. La frustrazione che ne risulta è un invito a non scrivere più test. Ma c'è una soluzione: testare tutto, o quasi.

Se scrivi codice senza test unitari stai "contraendo un debito", stai accumulando codice che alla fine diventerà ingestibile e pieno di bug. In questo senso la gran parte dei programmatori continua a fare debiti per pagare gli interessi dei debiti fatti in precedenza. Ma che vita è?

Come dovremmo testare?

Dovremmo usare un framework per scrivere test unitari, in modo che tutti i test possano poi girare in modo automatico. Il più famoso tra questi framework è JUnit, scritto per il linguaggio Java. Tutti i test, presi insieme, costituiscono una suite di test sull'intero sistema. Ciascun test in una suite dovrebbe avere un risultato booleano: o riesce, o fallisce. La suite stessa è booleana, perché fallisce quando almeno uno dei suoi test fallisce.

abbiamo scritto. Non possiamo mai essere sicuri al cento per cento che il nostro codice non contenga difetti, ma la presenza di una batteria completa di test unitari ci rasserena molto. Naturalmente questa sicurezza si paga: più della metà del codice che abbiamo scritto sono test. Ne sarà valsa la pena? Prima di giudicare, considerate due cose.

- **Primo:** il codice che abbiamo scritto è molto semplice, perché lo spazio tiranno ci impedisce di fare esempi davvero interessanti. In casi come questo i test unitari possono sembrare eccessivi. Siamo sicuri che molti di voi avranno pensato che una classe così banale non meritasse la fatica di scrivere tutti questi test. Ma naturalmente non tutto il codice è semplice come questo. E anche una classe semplice può contenere dei bug, come è capitato a noi il mese scorso. Quante volte vi è capitato di perdere ore di tempo alla ricerca di qualche bug "impossibile", per poi scoprire che si trattava di un errore banale in qualche punto del codice che consideravate semplicissimo? Quando il vostro programma sarà diventato abbastanza grosso da non poterlo più comprendere immediatamente con un solo sguardo, a quel punto sarete lieti del fatto che tutto il codice è coperto da test unitari, e che la semplice pressione di un tasto vi permette di sapere che tutto funziona correttamente.

- **Secondo**: nei nostri esempi abbiamo spiegato i test e il codice in grande dettaglio. Ma quando un programmatore ha preso il giusto ritmo, scrivere il codice e i test diventa un'operazione molto veloce. Tutto il codice di questo articolo e del precedente, test compresi, può essere scritto nell'arco di qualche minuto. Se un diavoletto sulla vostra spalla vi suggerisce di non perdere tempo con i test e di sbrigarvi a finire il codice, non fatevi ingannare. Probabilmente paghereste questa decisione, se non con la vostra anima, almeno con le ore perse inseguendo qualche bug.

Ma in questo articolo abbiamo filosofeggiato anche troppo. Vi diamo appuntamento al mese venturo con il terzo e ultimo articolo della nostra serie, che vi dimostrerà con un esempio pratico in che modo i test possono aiutarci a fare una cosa difficilissima: cambiare il codice che abbiamo già scritto senza introdurre bug. A presto!

Paolo Perrotta

L'esperto risponde...

Aggiungere controlli VB in maniera dinamica

Gentile Redazione, vi chiedo di fornirmi una risposta a questa mia esigenza. So che, con Visual Basic, si possono aggiungere controlli in fase di esecuzione del programma. Il fatto è che non so bene come si deve implementare un meccanismo del genere. Grazie e buon lavoro.

Giacomo

Aggiungere controlli ad una form, in maniera dinamica, cioè in fase di esecuzione (run-time), non è un'operazione molto complessa se si ha familiarità con la collezione dei controlli VB. Un uso tipico di questa collezione è il seguente:

```
Controls.Add "VB.TextBox", "FinestraTesto"
```

Il primo parametro riceve l'ID del controllo che si vuole aggiungere. Si può determinare il nome di un controllo effettuando una ricerca nel VB Object Browser. Il secondo parametro, invece, definisce il nome del nuovo controllo. La dichiarazione di cui sopra aggiunge una textbox chiamata *FinestraTesto* alla form corrente. Questo tipo di dichiarazione può essere utilizzato per aggiungere un controllo per qualsiasi tipo di oggetto contenitore, come un frame o una PictureBox, ad esempio. Per fare ciò, è sufficiente passare un riferimento a tale contenitore come terzo parametro della dichiarazione:

```
Controls.Add "VB.TextBox", "FinestraTesto",
                                      Frame1
```

È noto che, quando si aggiunge un controllo con questo metodo, VB lo mantiene invisibile per default. Per visualizzare il controllo, quindi, bisognerà impostare correttamente il parametro di visibilità nella procedura, subito dopo avere impostato i parametri per le dimensioni e la posizione del controllo all'interno del contenitore. Per effettuare una prova di funzionamento, all'interno di un progetto VB, basterà trascinare un pulsante e un frame nella form di default. In seguito, sarà sufficiente cliccare col tasto destro del mouse e selezionare la voce Visualizza codice. Nella finestra del codice, inserire la seguente procedura:

```
Private Sub Command1_Click()
Dim txtBox As TextBox
Set txtBox = Controls.Add("VB.TextBox", "
FinestraTesto", Frame1)
With txtBox
.Move 150, 240, 1500
.Visible = True
End With
End Sub
```

Premere il tasto *[F5]* e cliccare sul pulsante della form. VB aggiunge automaticamente una textbox al frame, collocandola esattamente nella posizione da noi impostata.

Ottimizzazione dei cicli for in Java

Gentile Redazione di ioProgrammo, tralasciando i complimenti stra-meritati, vorrei conoscere la vostra opinione in merito al fatto che i cicli for possono essere ottimizzati utilizzando gli operatori di incremento/decremento direttamente nel confronto. Vorrei sapere come sia possibile attuare questa tecnica o, comunque, come ottimizzare i loop.

Antonio

Prima di tutto, occorre essere sicuri che la variabile contatore utilizzata nel ciclo *for* sia una variabile *int* locale e non un'istanza o una variabile di una classe. A questo punto, possiamo ristrutturare il ciclo per aumentare le prestazioni, ottimizzando le comparazioni. Ad esempio, il ciclo:

```
for (int i = 0; i < max; i++)
```

può essere riscritto nella forma seguente:

```
for (int i = max; --i >= 0; )
```

che presenta due vantaggi. Innanzitutto, la comparazione avviene con una costante *(0)*, che è più veloce rispetto ad una comparazione con una variabile. In seguito, l'operatore di decremento utilizzato direttamente nel confronto è molto più veloce rispetto all'operazione di incremento. Ad esempio, possiamo utilizzare una struttura del genere per copiare gli elementi di un array in un altro, rendendo tale operazione molto più veloce…

Svegliare un thread dormiente

Gentile Redazione, vorrei sottoporvi un quesito tecnico. Ho da poco scoperto la programmazione mulithread in Java e devo dire che la trovo davvero interessante e ricca di "sfide" per uno sviluppatore. Ora mi trovo di fronte a un problema: vorrei riuscire a "svegliare" un thread in stato di sleep, prima che sia terminato il suo tempo di sleeping. È possibile? E se possibile, come si fa?

Daniele Lucarelli

Se l'oggetto *A* è in stato di sleep e l'oggetto *B* possiede il riferimento all'oggetto *A*, allora *B* può risvegliare *A* richiamando il metodo *interrupt()* sull'oggetto *A*:

```
A.interrupt()
```

Questa azione causerà la generazione di una eccezione *InterruptedException*, che dovrà dunque essere gestita nella classe in stato di sleep.
In realtà, volendo realizzare applicazioni robuste, è sempre bene gestire l'eccezione *InterruptedException*, stante l'eterogeneità degli ambienti in cui un'applicazione Java può essere eseguita.

Per contattarci:
e-mail: iopinbox@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

MozDev.org

http://www.mozdev.org/

Mozilla è un browser open-source di grande fama. Disponibile per la stragrande maggioranza delle piattaforme esistenti, unisce la completezza alla facilità d'uso. Il favore che sta incontrando, soprattutto tra la cerchia dei programmatori e degli appassionati, è motivato da numerosi fattori. Prima di tutto, Mozilla è un ottimo browser. Le pagine Web vengono visualizzate correttamente e velocemente, tutti i più importanti standard del W3C sono regolarmente supportati e, inoltre, è possibile navigare in piena sicurezza. Alcune particolari caratteristiche, come il *tabbed browsing*, il *cookie manager* e la possibilità di inibire le finestre pop-up non richieste, rendono Mozilla più appetibile di numerosi suoi concorrenti, primo fra i quali va annoverato Microsoft Internet Explorer. Se siete estranei al mondo di Mozilla, visitate la pagina all'indirizzo *http://www.xulplanet.com/ndeakin/arts/reasons.html*, che vi elencherà ben 101 motivi per cui Mozilla è da preferirsi ad Explorer. Si tratta, a dire il vero, di una lista di parte, stilata da uno degli sviluppatori del browser open-source, tuttavia condivisibile nella maggior parte dei suoi punti. Quanto detto sinora, ad ogni modo, è risaputo. Quello che non tutti sanno è che Mozilla, oltre ad essere un semplice browser, è anche una vera e propria piattaforma di sviluppo per applicazioni multipiattaforma orientate al Web. Quando Mozilla era ancora in beta testing, all'incirca un anno fa, trattai l'argomento su queste stesse pagine. Da allora, le cose si sono ulteriormente evolute. Il rilascio di una prima versione stabile del browser (la 1.0) ha portato all'ufficializzazione degli standard che sono dietro il suo framework. Programmare con Mozilla, in sostanza, significa realizzare add-on per il browser, ma non solo. La stessa piattaforma di sviluppo è impiegata per la creazione di altri software che incorporano il motore del browser, come Netscape 6 e 7, Galeon, Chimera e Phoenix. Esiste un buon libro, edito da O'Reilly, che spiega come realizzare applicazioni con il framework di Mozilla. Trovate informazioni sul testo all'indirizzo *http://www.oreilly.com/catalog/mozilla/*. Sviluppare applicazioni con Mozilla è relativamente semplice, giacché tutti i paradigmi di programmazione inclusi nel suo framework si basano su standard noti e recenti, sempre facilmente assimilabili. Le interfacce grafiche delle applicazioni, ad esempio, possono essere definite combinando l'uso di CSS e XML, nell'insieme nominato *XUL*. Proprio XML, infatti, ricopre un ruolo centrale nelle tecniche di sviluppo accettate dalla piattaforma. Le diverse funzionalità possono poi essere realizzate servendosi di semplice codice di scripting. *MozDev.org* è il punto di riferimento centrale per tutti gli sviluppatori di applicazioni per Mozilla. Il sito, come nel caso di numerosi altri progetti open-source, può essere sfruttato su due piani differenti. Per prima cosa, MozDev si rivolge agli utenti finali del browser. Chiunque, esperto o meno, può consultare il sito alla ricerca di add-on o software gratuiti e liberi, da impiegare a proprio vantaggio. L'archivio dei progetti ospitati è molto vasto. Si parte dalle skin per Mozilla e derivati, fino ad arrivare ad alcune tab molto specifiche per la sidebar di tutti i browser compatibili con gli standard del browser open-source. I singoli progetti possono essere navigati ed esplorati in maniere differenti. Il percorso più pratico, a mio avviso, è costituito dalla suddivisione in categorie del materiale esposto. Nel vasto elenco presentato da MozDev, grazie a questo meccanismo, diventa facile distinguere tra add-on, interfacce grafiche e browser alternativi, giochi, librerie di programmazione e molto altro ancora. La seconda modalità d'uso di MozDev si rivolge, come è lecito attendersi, agli sviluppatori. Avete realizzato un'applicazione o un add-on basato sul framework del browser? Bene, potrete renderlo fruibile alla comunità servendovi di MozDev. Gli strumenti messi a disposizione dai gestori del sito vi forniranno lo spazio e la visibilità necessari. Non sottovalutate questo aspetto! MozDev è uno dei principali punti di riferimento per lo sviluppo del celebre browser open-source. Diversi progetti pubblicati e diffusi da MozDev, originariamente sviluppati come semplici add-on opzionali, sono poi stati reputatati tanto validi ed utili da entrare a far parte della distribuzione di base di Mozilla. Dunque, se siete programmatori interessati a far parte della comunità di Mozilla, MozDev potrebbe essere la vostra rampa d'accesso in questa interessante realtà. All'infuori delle funzionalità offerte, il sito è ben organizzato, gradevole alla vista, sviluppato in osservanza dei più recenti standard. Per giunta, è facilmente usabile. Gli aggiornamenti sono frequenti ed i progetti ospitati numerosi ed ottimamente filtrati. Un bookmark importante per ogni amante della lucertola più famosa del Web.

Carlo Pelliccia